UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ANA SOFIA MOREIRA DE LIRA

Comparative Study of Quantum State Preparation Methods in Sparse Isometry Decomposition

Recife
2025

ANA SOFIA MOREIRA DE LIRA

Comparative Study of Quantum State Preparation Methods in Sparse Isometry Decomposition

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Multidisciplinar

Orientador (a): Adenilton José da Silva

Recife
2025

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Lira, Ana Sofia Moreira de.
Comparative Study of Quantum State Preparation Methods in Sparse Isometry Decomposition / Ana Sofia Moreira de Lira. - Recife, 2025.
81f.: il.

Dissertação (Mestrado) - Universidade Federal de Pernambuco, Centro de Informática, Programa de Pós-graduação em Ciência da Computação, 2025.
Orientação: Adenilton José da Silva.
Inclui referências e apêndices.

1. Householder decomposition; 2. Isometry; 3. Sparse isometry; 4. State preparation. I. Silva, Adenilton José da. II. Título.

UFPE-Biblioteca Central

Ana Sofia Moreira de Lira

“Comparative Study of Quantum State Preparation Methods in Sparse Isometry Decomposition”

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Inteligência Computacional

Aprovado em: 06/02/2025.

BANCA EXAMINADORA

_____

Prof. Dr. Stefan Michael Blawid
Centro de Informática / UFPE

_____

Profa. Dra. Nadja Kolb Bernardes
Departamento de Física / UFPE

_____

Prof. Dr. Israel Ferraz de Araújo
Yonsei University

_____

Prof. Dr. Adenilton José da Silva
Centro de Informática / UFPE
(orientador)

Dedico a todos que, de alguma forma, acreditaram em mim.

ACKNOWLEDGEMENTS

Aos professores que fizeram parte da minha vida, agradeço por terem me guiado na trajetória que me trouxe até aqui.

Aos meus colegas de laboratório, por estarem presentes nos meus dias e tornar a rotina mais divertida.

Aos companheiros do grupo de pesquisa, pelo aprendizado compartilhado.

Aos meus amigos, cujo apoio emocional foi fundamental.

À minha terapeuta, por me oferecer um espaço seguro para existir e crescer, e à terapia, por me dar ferramentas para continuar.

À minha mãe, Lilian, por todos os sacrifícios feitos por mim. Aos meus irmãos, Saimon e Ícaro, pelas memórias compartilhadas.

Ao professor e orientador Adenilton, do qual o suporte foi vital para o desenvolvimento desta pesquisa e cuja postura e entusiasmo me motivam a continuar estudando.

Este trabalho foi financiado pela CAPES e pelo CNPq, cujos apoios foram essenciais para a realização desta pesquisa.

Meanwhile the wild geese, high in the clean blue air,
are heading home again.
Whoever you are, no matter how lonely,
the world offers itself to your imagination,
calls to you like the wild geese, harsh and exciting –
over and over announcing your place
in the family of things. (OLIVER, 1986).

ABSTRACT

Initializing an isometry in a quantum circuit is a fundamental yet challenging task, especially when aiming for efficient state preparation and resource optimization. In this work, we present a comparative analysis of the Householder Decomposition for isometry implementation, leveraging three distinct state preparation methods starting from the original Pivot method and extending the study to two additional strategies: Merge and Low Rank. The study investigates how each method impacts the performance of the isometry decomposition, focusing on key metrics such as CNOT gate count and circuit depth. To provide a comprehensive evaluation, we examine variations in matrix size and sparsity levels, capturing the effects of structural complexity on resource requirements.

Our results reveal that the Merge state preparation method generally outperforms the other two approaches, particularly in terms of scalability and gate efficiency. Building upon these findings, we further compare the best-performing method, Merge, with the state-of-the-art isometry decomposition implementation available in Qiskit, a widely used quantum computing framework. The analysis demonstrates that, for isometries involving up to 6 qubits, Qiskit's implementation exhibits superior performance. However, beyond this threshold, our proposed decomposition method proves more effective, especially for highly sparse isometries or those characterized by a smaller number of columns.

This work highlights the potential for optimizing isometry decompositions in scenarios where sparsity and structural constraints are critical factors. These findings contribute to advancing state preparation techniques and offer insights into improving the efficiency of quantum circuits for applications in quantum information processing.

Keywords: Householder Decomposition; isometry; sparse isometry; state preparation.

RESUMO

Inicializar uma isometria em um circuito quântico é uma tarefa fundamental, porém desafiadora, especialmente quando se busca uma preparação de estado eficiente e otimização de recursos. Neste trabalho, apresentamos uma análise comparativa da Decomposição de Householder para a implementação de isometrias, explorando três métodos distintos de preparação de estados — iniciando pelo método original, Pivot, e estendendo o estudo para duas estratégias adicionais: Merge e Low Rank. O estudo investiga como cada método impacta o desempenho da decomposição de isometrias, com foco em métricas-chave, como o número de portas CNOT e a profundidade do circuito. Para fornecer uma avaliação abrangente, analisamos variações no tamanho da matriz e nos níveis de esparsidade, capturando os efeitos da complexidade estrutural nos requisitos de recursos.

Nossos resultados mostram que o método de preparação de estado Merge geralmente supera as outras duas abordagens, especialmente em termos de escalabilidade e eficiência em portas lógicas. Com base nesses resultados, comparamos o método com melhor desempenho, Merge, com a implementação de decomposição de isometrias disponível no Qiskit, uma das bibliotecas de computação quântica mais amplamente utilizadas. A análise demonstra que, para isometrias envolvendo até 6 qubits, a implementação do Qiskit apresenta desempenho superior. No entanto, além desse limite, o método proposto revela-se mais eficiente, particularmente para isometrias altamente esparsas ou caracterizadas por um número reduzido de colunas.

Este trabalho destaca o potencial para otimizar decomposições de isometrias em cenários onde a esparsidade e restrições estruturais são fatores críticos. Esses resultados contribuem para o avanço das técnicas de preparação de estados e oferecem insights sobre como melhorar a eficiência de circuitos quânticos em aplicações de processamento de informação quântica.

Palavras-chaves: Decomposição Householder; isometria; isometria esparsa; preparação de estados.

LIST OF FIGURES

Figure 1 – Illustrative image showing the width and depth of an illustrative circuit. . . 20

Figure 2 – Illustration of a possible sparse quantum state. The empty blocks represent zero amplitudes, while the colored blocks indicate non-zero amplitudes, with their corresponding values displayed below the arrows. This state can be mathematically expressed as: $|\psi\rangle = \frac{1}{\sqrt{2}}|011\rangle + \frac{1}{\sqrt{2}}|101\rangle$. 20

Figure 3 – Representation of a controlled unitary gate acting on a single qubit, with a closed control. The gate U is activated only when the control qubit is in the state $|1\rangle$. 26

Figure 4 – Representation of a controlled unitary gate acting on a single qubit, with an open control. The gate U is activated only when the control qubit is in the state $|0\rangle$. This configuration is equivalent to placing a NOT gate (X gate) before and after the control qubit. 27

Figure 5 – Quantum circuit representing the application of a CNOT gate. 27

Figure 6 – Decomposition of a multicontrolled unitary gate of five qubits, with $\{a_1, a_2, a_3, a_4\}$ acting as controls and $\{a_5\}$ acting as the target. The R_x gates are represented by boxes labeled with their respective angles in the form π/k, where $k \in N$. 29

Figure 7 – Illustration of the first step of Algorithm 2, where two suitable basis states are identified and merged to form a new state, reducing the system by one basis state and allowing the process to repeat. 32

Figure 8 – Illustration of Algorithm 1. The first step is the pre-processing of classical data D. After that, the circuit is started as an empty state with one ancilla. After that, each iteration of the process is responsible for initializing the binary pattern and its corresponding amplitudes, progressing from the simplest to the most complex, enabling system initialization. 35

Figure 9 – Illustration of Algorithm 1, showing the initial state $|v\rangle$ and the final state $|u\rangle$. The block T highlights the target qubits, while components in purple have amplitudes of $\frac{1}{\sqrt{2}}$. All other components have zero amplitude. 37

Figure 10 – Illustration of the Low-Rank Algorithm flow. The classical stage performs the Schmidt decomposition. The quantum state initializes a quantum state in a specific form, followed by the addition of CNOT gates. Finally, unitary gates are applied to generate the desired state. 39

Figure 11 – Example of an isometry in a quantum circuit. 41

Figure 12 – Geometric representation of the Householder reflection. 42

Figure 13 – Controlled $-Z$ operation in a 3-qubit circuit. 45

Figure 14 – Illustration of the Householder decomposition. The process begins with an isometry V, composed by zeros and arbitrary complex numbers $*$, where the first step involves applying the Householder reflection H_{v_0}, transforming V into a new isometry. After at most m operations, the isometry is reduced to a diagonal matrix. 46
Figure 15 – Quantum circuit illustrating the first step of a possible Householder decomposition. In this step, a state preparation is applied, followed by a controlled $-Z$ gate acting on the third qubit, and concluding with the application of the transposed conjugate of the state preparation. 46
Figure 16 – Controlled U operation in a n-qubit circuit. It is a subcircuit used to implement the Householder Reflection with respect to $|v\rangle$ algorithm (see Section 3.1.2), where a single-qubit phase gate that applies a phase ϕ to the $|0\rangle$ component only. 49
Figure 17 – Quantum circuit of the Generalized Householder Reflection. The inverse of a state preparation is applied, followed by a controlled unitary gate U acting on the target qubit, and concluding with the application of the state preparation. 50
Figure 18 – Illustration of a Householder Decomposition circuit applied to n qubits, processing m columns of the matrix. The operation SP_{v_i} represents the state preparation at step i, while $SP_{v_i}^{\dagger}$ denotes its Hermitian adjoint (conjugate transpose). The gate U_i corresponds to the multi-controlled unitary operation associated with the vector $|v_i\rangle$. 51
Figure 19 – Number of qubits versus number of CNOT gates for varying $1 \leq s \leq 4$ with $m = 1$ fixed. The curves correspond to different state preparation methods applied to Householder Decompositions—green represents Pivot, blue represents Low Rank, and orange represents Merge 55
Figure 20 – Number of qubits versus number of CNOT gates for varying $1 \leq m \leq 4$ with $s = 1$ fixed. The curves correspond to different state preparation methods applied to Householder Decompositions—green represents Pivot, blue represents Low Rank, and orange represents Merge. 55
Figure 21 – Number of Qubits vs Number of Cnots for varying $1 \leq s \leq 4$ and $1 \leq s \leq 3$. The curves correspond to isometry decompositions – red represents the Column-by-column isometry method and orange represents The Householder Merge Decomposition. 56
Figure 22 – Number of qubits versus Depth count for varying $1 \leq s \leq 4$ with $m = 1$ fixed. The curves correspond to different state preparation methods applied to Householder Decompositions—green represents Pivot, blue represents Low Rank , and orange represents Merge. 58

Figure 23 – Number of qubits versus Depth count for varying $1 \leq m \leq 4$ with $s = 1$ fixed. The curves correspond to different state preparation methods applied to Householder Decompositions—green represents Pivot, blue represents Low Rank, and orange represents Merge. 59
Figure 24 – Number of Qubits vs Depth count for varying $1 \leq s \leq 4$ and $1 \leq s \leq 3$. The curves correspond to isometry decompositions – red represents the Qiskit isometry method and orange represents The Householder Merge Decomposition. 60

LIST OF SYMBOLS

n	Total number of qubits
m	Number of active input qubits of an isometry
s	Sparsity metric
$\|v\rangle$	State vector
ϕ	Phase
H_v^ϕ	Generalized Householder Reflection with respect to $\|v\rangle$
H_0^ϕ	Householder Reflection with respect to $\|0\rangle$
SP_v	State preparation applied to state $\|v\rangle$
C_n	Controlled operation with n controls

LIST OF ALGORITHMS

1 H_0^ϕ or Householder Reflection with respect to $|0\rangle$ 48
2 H_v^ϕ or Householder Reflection with respect to $|v\rangle$ 49
3 Householder Decomposition for Isometry 52

CONTENTS

1 INTRODUCTION . . . 18
1.1 MOTIVATION . . . 19
1.2 OBJECTIVES . . . 21
1.3 ORGANIZATION OF THE DISSERTATION . . . 22
2 THEORETICAL BACKGROUND . . . 23
2.1 QUANTUM COMPUTING . . . 23
2.2 DECOMPOSITION OF MULTI-CONTROLLED UNITARY GATES . . . 28
2.2.1 Linear-depth quantum circuits for multiqubit controlled gates . . . 28
2.3 STATE PREPARATION . . . 29
2.3.1 Merge Algorithm . . . 30
2.3.2 CVO-QRAM Algorithm . . . 33
2.3.3 Pivot Algorithm . . . 36
2.3.4 Low-Rank Algorithm . . . 38
2.4 ISOMETRY . . . 40
2.5 DECOMPOSITION OF SPARSE ISOMETRIES . . . 41
2.5.1 Householder Decomposition . . . 42
3 METHODOLOGY . . . 47
3.1 HOUSEHOLDER DECOMPOSITION ALGORITHM . . . 47
3.1.1 Implementation of Householder Reflection with Respect to the State $|0\rangle$ . . . 48
3.1.2 Implementation of Householder Reflection with Respect to the state $|v\rangle$ . . . 49
3.1.3 Implementation of Householder Decomposition . . . 50
3.2 SPARSE MATRICES . . . 52
4 EXPERIMENTS AND RESULTS . . . 54
4.1 CNOT ANALYSIS . . . 54
4.2 DEPTH ANALYSIS . . . 57
4.3 CHALLENGES IN APPLYING HOUSEHOLDER DECOMPOSITIONS . . . 60
5 FINAL CONSIDERATIONS . . . 62
REFERENCES . . . 64
APPENDIX.A . . . 68
MERGE STATE PREPARATION . . . 68
APPENDIX.B . . . 74
CVO-QRAM STATE PREPARATION . . . 74
APPENDIX.C . . . 79
PIVOT STATE PREPARATION . . . 79
APPENDIX.D . . . 82

LOW-RANK STATE PREPARATION 82

1 INTRODUCTION

Quantum computation is the study of the information processing tasks that can be accomplished using quantum mechanical systems (NIELSEN; CHUANG, 2000), that is, advanced machines inspired by the principles of quantum mechanics: superposition, entanglement, and interference (SCHNEIDER; SMALLEY, 2024). These principles allow qubits (basic elements of quantum information) to exist in multiple simultaneous states that can be processed in parallel. A circuit, the fundamental building block of quantum computers, consists of combinations of sequential and parallel gate operations (YANOFSKY; MANNUCCI; MANNUCCI, 2008). These are represented by wires and elementary gates that carry and manipulate data.

The history of quantum computation dates back to the 1980s, when Paul Benioff, an American physicist, described the construction of a quantum mechanical model of computers represented by a Turing machine (BENIOFF, 1980). The idea of a Universal Quantum Computer came in 1985, when David Deustch, a British physicist, published the idea of a device that would be capable of efficiently simulating any physical system (DEUTSCH, 1985). This idea came three years after Nobel-winning physicist Richard Feynman proposed the first use of quantum computation to simulate physical processes, suggesting that classical computers would struggle to simulate quantum mechanics in an efficient way (FEYNMAN, 1982). After almost a decade, Peter Shor demonstrated that a quantum computer could efficiently find discrete logarithms and factor integers, which can be used to break several proposed cryptosystems, such as the RSA system (GIDNEY; EKERå, 2021; RIVEST; SHAMIR; ADLEMAN, 1978). This algorithm is now referred to as "Shor's Algorithm" (SHOR, 1994). In 1995, Lov Grover presented another important algorithm, known as "Grover's algorithm", responsible for searching unsorted data, with a quadratic acceleration compared to its best classical version (GROVER, 1996).

Recently, a new wave of quantum algorithms have been proposed in defiance of the narrow view of what a quantum computer would be useful for, showing how quantum computers might be used to provide exponential gain compared to its classical adversary (AARONSON, 2015). Some practical applications include machine learning, clustering, classification, and finding patterns in huge amounts of data (AARONSON, 2015; BIAMONTE et al., 2017; SCHULD; SINAYSKIY; PETRUCCIONE, 2014; SCHULD; PETRUCCIONE, 2018). An important milestone in this area was achieved in 2019, when Google announced it had reached quantum supremacy with its Sycamore processor by completing a complex random circuit sampling task in minutes, which would take thousands of years to complete on the best supercomputer existing at the time (ARUTE; AL., 2019). Although many advantages are expected in the fields of research and industry, there are some challenges that prevent these applications from being implemented, like noise, limited connectivity, and constraints on qubit count (width) and circuit depth (number of operation layers) (LEYMANN; BARZEN, 2020; AARONSON, 2015).

An important component in the operation of these algorithms is the encoding of classical data into a quantum state (AARONSON, 2015; BIAMONTE et al., 2017; RATH; DATE, 2024). Encoding classical data requires converting it to vector form, after which it is translated into operations performed on a quantum computer, usually encoded in a quantum circuit. The execution of this circuit creates the initial desired quantum state (NIELSEN; CHUANG, 2000; YANOFSKY; MANNUCCI; MANNUCCI, 2008).

The process of encoding classical data into a quantum state is called **Quantum State Preparation Algorithm** (QSPA). This step is necessary because quantum algorithms rely on quantum bits, or qubits, for computation. While a classical bit is limited to the states 0 or 1, a qubit can be in a quantum superposition of the basis states $|0\rangle$ and $|1\rangle$ (SAKURAI; NAPOLITANO, 2020; EISBERG; RESNICK, 1985). The complexity of initializing a quantum state plays a crucial role in estimating the cost of QSPA, as it can potentially overcome the advantages of quantum speedup (RATH; DATE, 2024; VERAS et al., 2021).

1.1 MOTIVATION

Many companies and research institutions, including IBM, Google, Microsoft, and Amazon[1], are actively investing in the development of quantum devices. Despite that, the actual quantum devices are noisy and unreliable. These devices, known as $NISQ$ (which stands for *Noisy Intermediate-Scale Quantum*), are limited by the amount of qubits, the connectivity between them, and the presence of noise in their operations (PRESKILL, 2018). This leads to the difficulty to execute circuits with large depths — corresponding to how many layers of quantum gates need to be applied in sequence — and width — corresponding to the amount of qubits used simultaneously in a circuit (PRESKILL, 2018). Figure 1 provides an example of the depth and width of an illustrative circuit.

As previously discussed, initializing a quantum state can be a complex and time-consuming process (RATH; DATE, 2024; VERAS et al., 2021). In his well-established book Introduction to Algorithms, computer scientist Thomas Cormen (CORMEN et al., 2009) explores the growth of functions in algorithms. This concept of growth provides a straightforward characterization of an algorithm's efficiency. To compare growth rates, computer scientists commonly use three notations: O (big O), Ω (big Omega), and Θ (big Theta). These notations describe the asymptotic behavior of a function in relation to the input size. For example, if N is the input lenght of an algorithm, $g(N)$ is a function that represents a potential growth rate or

[1] To access the official channels of these companies:

- IBM: https://www.ibm.com/quantum
- Google: https://quantumai.google/
- Microsoft: https://quantum.microsoft.com/
- Amazon: https://aws.amazon.com/pt/braket/

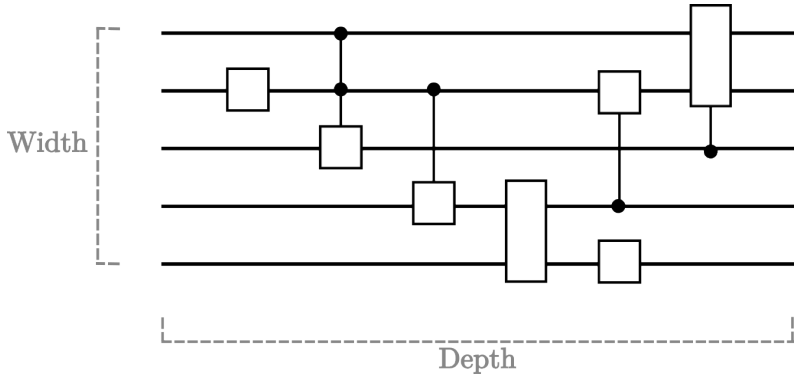


Figure 1 – Illustrative image showing the width and depth of an illustrative circuit.

complexity. In the case of Big O, $O(g(N))$ represents the upper bound of the algorithm's growth rate, indicating that the function will not grow faster than $g(N)$ multiplied by a constant for sufficiently large N. In contrast, $\Omega(g(N))$ represents the lower bound, meaning that the function will grow at least as fast as $g(N)$ for large N. Finally, $\Theta(g(N))$ indicates that the function grows at the same rate as $g(N)$ both asymptotically from above and below, meaning the function's growth is bounded both above and below by $g(N)$ within constant factors for large N. In all cases, $g(N)$ serves as a reference function that characterizes the growth rate of the algorithm or function being analyzed. In this work, we will focus on the big O notation.

In the context of quantum computing, these same principles of complexity analysis are applied to quantum algorithms, with particular emphasis on the structure and resources required by quantum circuits. Depth and width are typically used as parameters to determine the efficiency of a circuit (NIELSEN; CHUANG, 2000). The worst-case complexity of Quantum State Preparation is exponential in relation to the number of qubits (SHENDE; BULLOCK; MARKOV, 2005). In comparison, a hypothetical best-case scenario would have logarithmic complexity, although this has not yet been achieved.

An alternative to get around the problems of the use of NISQ devices is to find solutions that align with them. In algorithms where Quantum State Preparation is a essential for achieving quantum speedup (BIAMONTE et al., 2017), the motivation for improving its efficiency becomes even more evident (AARONSON, 2015; RøNNOW et al., 2014). Hence, the importance of finding strategies to facilitate the use of such devices.

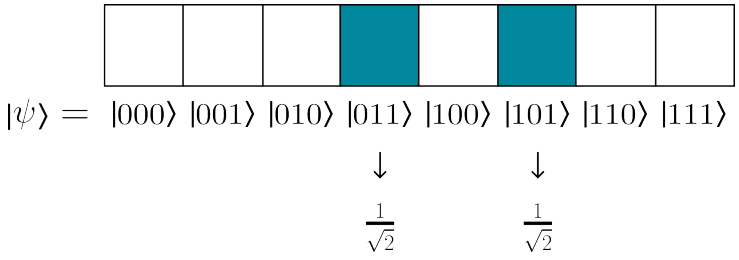


Figure 2 – Illustration of a possible sparse quantum state. The empty blocks represent zero amplitudes, while the colored blocks indicate non-zero amplitudes, with their corresponding values displayed below the arrows. This state can be mathematically expressed as: $|\psi\rangle = \frac{1}{\sqrt{2}}|011\rangle + \frac{1}{\sqrt{2}}|101\rangle$.

Not all quantum states are created equal. Quantum speedup (BIAMONTE et al., 2017) can be more effectively achieved when the initial states for algorithms are easier to prepare. For example, if a state is considered sparse, meaning it has many zero-amplitude inputs (MALVETTI; ITEN; COLBECK, 2021; VERAS et al., 2021; GLEINIG; HOEFLER, 2021) as shown in Figure 2, it should be easier to initialize on a quantum computer.

1.2 OBJECTIVES

The main objective of this work is to perform a comparative analysis of various state preparation methods applied within the **Householder Decomposition** framework for isometry decomposition, as introduced by Malvetti et al. (MALVETTI; ITEN; COLBECK, 2021). The original work employs the Pivot-based state preparation as its core strategy. Here, we extend that analysis by exploring two alternative approaches – Merge and Low Rank – and compare them against the original method. Our investigation focuses on how these different techniques respond to specific structural features of the target isometry, such as its sparsity and number of columns. These characteristics play a crucial role in determining the circuit's complexity and resource requirements, making this comparison particularly relevant for the design of scalable and resource-efficient quantum algorithms.

Following this analysis, we evaluate the performance of each state preparation method based on two key metrics: the number of CNOT gates and the circuit depth. The number of CNOT gates is particularly relevant because two-qubit operations, such as the CNOT, are significantly more challenging to implement than single-qubit gates. They require an additional degree of freedom and tighter control over qubit interactions, which increases the chances of introducing errors. As a result, CNOT gates are a major source of decoherence in quantum circuits (ITEN et al., 2016).

Circuit depth, on the other hand, measures the number of gate layers that must be executed sequentially (NIELSEN; CHUANG, 2000). This is a crucial parameter, especially in the context of NISQ devices, which have limited coherence times (AARONSON, 2015). A deeper circuit takes longer to execute, increasing the risk that qubits will lose their quantum state before the computation finishes. Therefore, minimizing circuit depth helps ensure the circuit remains within the physical limits of current quantum hardware.

Finally, we benchmark the most efficient state preparation approach against the standard implementation available in Qiskit, one of the most widely used open-source quantum computing libraries (ALEKSANDROWICZ et al., 2019). At the time of this work, it features the Column-by-column decomposition (ITEN et al., 2016).

1.3 ORGANIZATION OF THE DISSERTATION

The remainder of this thesis is structured as follows:

- **Chapter 2:** Provides the theoretical foundation necessary for this work, including an overview of basic quantum computing concepts, the decomposition of multi-controlled unitary gates, state preparation techniques, isometries, and the Householder Decomposition.
- **Chapter 3:** Describes the methodology used in this study, detailing the implementation of the algorithms and the composition of the corresponding quantum circuits.
- **Chapter 4:** Presents the experimental procedures and key results, accompanied by a detailed analysis of the collected data. In addition, this chapter discusses the primary challenges encountered throughout the work.
- **Chapter 5:** Concludes the thesis by revisiting the objectives and summarizing the main findings. It also outlines potential directions for future research.

2 THEORETICAL BACKGROUND

2.1 QUANTUM COMPUTING

Quantum computing is a novel approach to computation that leverages the principles of quantum mechanics (YANOFSKY; MANNUCCI; MANNUCCI, 2008; NIELSEN; CHUANG, 2000). When studying quantum computation, it is common to make an analogy to its classical counterpart.

In classical computing, information is stored and processed using bits, which can take two different values: 0 or 1 (MACKENZIE, 1980). Mathematically, a classical bit can take one of either two values:

$$BIT = \{0, 1\} \tag{2.1}$$

This approach is simple and efficient for many tasks used in our common computers, but imposes limitations when dealing with problems that require massive parallelism or probabilistic behavior (PRESKILL, 2018).

Just as classical bits form the foundation of classical computing, qubits serve as the basic units of information in quantum computers (NIELSEN; CHUANG, 2000). In simpler terms, you can think of classical bits as straightforward on/off switches, while qubits behave more like spinning coins—until measured, they exist in a state that is both heads and tails at the same time (WHURLEY; SMITH, 2023). They are usually represented using Dirac's notation, with $|0\rangle$ as the quantum analog for 0 and $|1\rangle$ as the quantum analog for 1. They can also be represented using matrix notation:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.2}$$

Qubits are unique because they explore the principles of quantum mechanics, which allows them to exist in a superposition of $|0\rangle$ and $|1\rangle$ simultaneously. Mathematically, we represent a state $|v\rangle$ in superposition as a two-dimensional vector that is a linear combination of $|0\rangle$ and $|1\rangle$:

$$|v\rangle = c_0 |0\rangle + c_1 |1\rangle \tag{2.3}$$

where c_0 and c_1 are the complex coefficients that satisfy the normalization condition:

$$|c_0|^2 + |c_1|^2 = 1 \tag{2.4}$$

Equation 2.4 indicates that the qubit exists in both states $|0\rangle$ and $|1\rangle$, where $|c_0|^2$ is the probability of measuring $|0\rangle$ and $|c_1|^2$ is the probability of measuring $|1\rangle$. It is the property of superposition that gives quantum computers their power of parallel computing.

We can also combine multiple qubits to represent more complex states, which are referred to as *basis states*. They can be represented as the tensor product of individual qubit states. In general, any n-qubit basis state $|x_1 x_2 \dots x_n\rangle$ corresponds to the tensor product $|x_1\rangle \otimes |x_2\rangle \otimes \dots \otimes |x_n\rangle$, where each $x_i \in \{0, 1\}$. We can also simplify the representation of a multiple qubit state by simply writing $|b_i\rangle^{\otimes n}$, which denotes a tensor product of n qubits all initialized in the state $|b_i\rangle$. This shorthand is commonly used to represent the initial state $|0\rangle^{\otimes n}$ of a quantum register before any gates are applied, and it corresponds to the full basis state $|00\dots0\rangle$ with n zeros.

Accordingly, we define basis states as follows:

Definition 1 *Basis states of a qubit system are a set of 2^n orthonormal vectors. A general state $|v\rangle$ of a n-qubit quantum system is a normalized linear combination of basis states:*

$$|v\rangle = \sum_{x \in \{0,1\}^n} c_x |x\rangle , \sum_{x \in \{0,1\}^n} |c_x|^2 = 1 \tag{2.5}$$

where c_x are the amplitudes of the basis state $|x\rangle$ (GLEINIG; HOEFLER, 2021).

In general, describing a quantum state requires describing all c_x amplitudes. For these states, we define the following properties:

Definition 2 *Letting $S \subset \{0, 1\}^n$ denote the set of basis states with nonzero coefficients. A quantum state is called **dense** if*

$$|v\rangle = \sum_{x \in S} c_x |x\rangle , \tag{2.6}$$

where $|S| \approx 2^n$, that is, when most of the basis states have nonzero coefficients.

Definition 3 *Letting $S \subset \{0, 1\}^n$ denote the set of basis states with nonzero coefficients. A quantum state is called **sparse** if*

$$|v\rangle = \sum_{x \in S} c_x |x\rangle , \tag{2.7}$$

where $|S| \ll 2^n$, that is, when only a small amount of bases states have nonzero coefficients (GLEINIG; HOEFLER, 2021).

To manipulate the basis states, we use quantum operators – also called quantum gates. They are mathematically represented as unitary matrices that act on the state vector of a qubit or a set of qubits. These gates transform the quantum state while preserving the total probability, as required by the principles of quantum mechanics. For example, the Hadamard gate H, represented by the matrix:

- **H gate:**

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.8}$$

This gate creates a superposition state when applied to a qubit in the $|0\rangle$ or $|1\rangle$ state:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Other basic gates, called Pauli Gates, consist of single qubit operations represented as:

- **X gate**:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.9}$$

Swaps the $|0\rangle$ and $|1\rangle$ states:

$$X|0\rangle = |1\rangle, \quad X|1\rangle = |0\rangle$$

- **Y gate**:

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{2.10}$$

Introduces a complex phase during the state transition:

$$Y|0\rangle = i|1\rangle, \quad Y|1\rangle = -i|0\rangle$$

- **Z gate**:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.11}$$

This gate flips the phase of the $|1\rangle$ state while leaving $|0\rangle$ unchanged:

$$Z|0\rangle = |0\rangle, \quad Z|1\rangle = -|1\rangle$$

Another commonly used set of gates is the rotation gates, which introduce an arbitrary angle θ into the circuit, allowing for more flexibility in manipulating quantum states. They are particularly useful for changing the phase or amplitude of quantum states and are implemented through mathematical rotations, which are used to represent unitary operators that act on a single qubit.

- R_x **gate:**

$$R_x = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \tag{2.12}$$

In a quantum circuit, this performs a unitary operation on a single qubit, rotating the state of the qubit by the angle θ around the x-axis of the Bloch sphere.

- R_y **gate:**

$$R_y = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \tag{2.13}$$

This gate rotates a qubit's state by an angle θ around the y-axis of the Bloch sphere.

- R_z **gate:**

$$R_z = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix} \tag{2.14}$$

This gate is used to rotate the phase of the qubit's state without affecting its amplitude. The R_z gate modifies the qubit's state by adding a phase factor, where the phase is dependent on the angle θ.

Another common type of quantum gate is the controlled gate, which introduces conditional logic into quantum circuits. Unlike the gates discussed earlier, controlled gates operate based on the state of one or more designated **control qubits**. An operation U, which can be any unitary transformation, is applied to a **target qubit** only if the control qubits satisfy a specific condition, such as being in the state $|1\rangle$ or $|0\rangle$. Figure 3 illustrates a circuit for a controlled unitary gate with a closed control, meaning that the unitary gate will only act if the control qubit is in the state $|1\rangle$. In contrast, Figure 4 shows a circuit for a controlled unitary gate with an open control, which means that the unitary gate will only act if the control qubit is in the state $|0\rangle$.

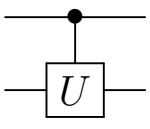


Figure 3 – Representation of a controlled unitary gate acting on a single qubit, with a closed control. The gate U is activated only when the control qubit is in the state $|1\rangle$.

The most used example of a controlled gate is the Controlled-NOT (CNOT) gate. This gate flips the state of the target qubit if the control qubit is in the state $|1\rangle$. Mathematically, the CNOT gate acting on two consecutive qubits is represented by the matrix:

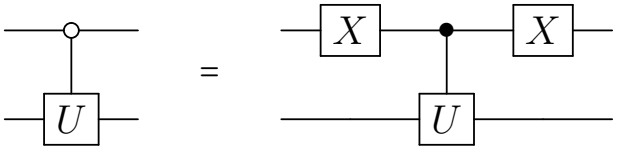


Figure 4 – Representation of a controlled unitary gate acting on a single qubit, with an open control. The gate U is activated only when the control qubit is in the state $|0\rangle$. This configuration is equivalent to placing a NOT gate (X gate) before and after the control qubit.

- **CNOT gate:**

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.15}$$

It works by checking if the control qubit is $|0\rangle$ or $|1\rangle$. If it is in the state $|0\rangle$, the target remains unchanged. If it is in the state $|1\rangle$, the target is flipped from $|0\rangle$ to $|1\rangle$ or vice versa. In a circuit, it is represented as seen in the figure 5:

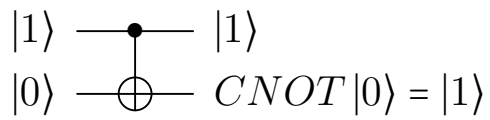
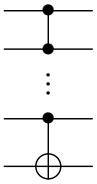


Figure 5 – Quantum circuit representing the application of a CNOT gate.

If a controlled gate has multiple control qubits, it is referred to as a multi-controlled gate. One special case of the Multi-controlled gate is when the unitary gate is the operation NOT. For two control qubits, the gate is known as a Toffoli gate (TOFFOLI, 1980). For three or more controls, the multi-controlled NOT (or multi-controlled Toffoli) gate is represented as:

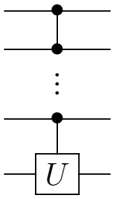
- **Multi-controlled Toffoli gate:**



Similar to multi-controlled gates, the Toffoli gate can be implemented in various ways; however, this is not the primary focus of this work.

The Toffoli gate can be generalized for an arbitrary unitary operation U, and is represented as:

- **Multi-controlled gate:**



There are different ways to implement a multi-controlled gates, but we will focus on a decomposition that presents a linear depth (see Section 2.2).

In addition to the gates previously introduced, there exists a set of gates known as "universal gates". These gates are called universal because any unitary operation can be approximated to arbitrary accuracy using only combinations of them. A common universal gate set includes the Hadamard, Phase, CNOT, and $\pi/8$ (T) gates. To complete this universal set, we now present the following gates:

- **Phase gate (S):**

$$S = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \tag{2.16}$$

Introduces a phase shift to the $|1\rangle$ component of a qubit state, leaving the $|0\rangle$ component unchanged.

$$S|0\rangle = |0\rangle, \quad S|1\rangle = i|1\rangle$$

- $\pi/8$ **gate (T):**

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{\sqrt{2}}(1+i) \end{bmatrix} \tag{2.17}$$

It adds a phase of $\pi/4$ radians to the $|1\rangle$ component of a qubit, while leaving the $|0\rangle$ component unchanged:

$$T|0\rangle = |0\rangle, \quad T|1\rangle = e^{i\frac{\pi}{4}}|1\rangle$$

2.2 DECOMPOSITION OF MULTI-CONTROLLED UNITARY GATES

2.2.1 Linear-depth quantum circuits for multiqubit controlled gates

The work presented by (SILVA; PARK, 2022) aims to decompose multi-controlled gates, denoted as C_nU, where a single qubit unitary operator U is applied to a target qubit conditioned

on the state of n control qubits. The circuit is structured in a pyramid-like shape and uses rotation gates R_x, which allows for an implementation of linear depth $O(n)$ and a quadratic cost $O(n^2)$ of CNOT gates.

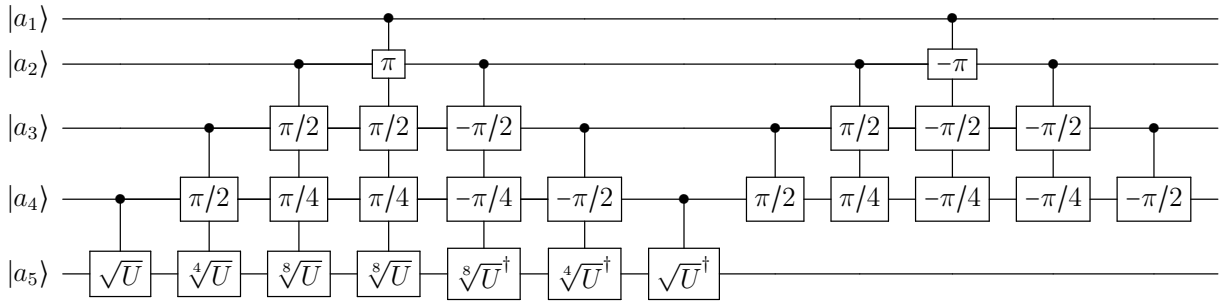


Figure 6 – Decomposition of a multicontrolled unitary gate of five qubits, with $\{a_1, a_2, a_3, a_4\}$ acting as controls and $\{a_5\}$ acting as the target. The R_x gates are represented by boxes labeled with their respective angles in the form π/k, where $k \in N$.

This method uses controlled rotations R_x, represented as boxes with their respective angles π/k, where $k \in N$, as shown in Figure 6. The rotations are applied to all qubits, except the first, which acts solely as a control. The total number of control qubits is $k = n-1$. In addition, the approach utilizes 2^j-th roots of the unitary operator U, where $j \in \{1, 2, \cdots, k-1\}$.

The main advantage of this circuit is that it enables multiple operations to be executed in parallel, provided they do not act on the same target qubit. For instance, the first layer consists of the unitary gate $\sqrt{U}$ applied to qubit $|a_5\rangle$, controlled by qubit $|a_4\rangle$. The second layer applies the gate $\sqrt[4]{U}$ to the same target qubit $|a_5\rangle$, again controlled by $|a_4\rangle$. Parallelism becomes more evident in the third layer, which includes a rotation gate $R_x(\pi/2)$ acting on qubit $|a_4\rangle$, controlled by $|a_3\rangle$, and simultaneously the gate $\sqrt[8]{U}$ applied to $|a_5\rangle$, controlled by qubit $|a_2\rangle$. This pattern continues, following a diagonal distribution on the pyramid shape.

As illustrated in Figure 6, for a circuit with 4 controls, notice that if any of the control qubits doesn't attend the required criteria (i.e., is in the state $|0\rangle$ rather than $|1\rangle$), the operator U is not applied to the target qubit. This happens because the controlled R_x rotations (represented by gates with angles of the shape π/j) effectively cancel each other out. In this way, the structure of the decomposition ensures that the operation is applied only when all control qubits are in the desired state.

2.3 STATE PREPARATION

When a task requires loading classical data into a quantum computer, the first step is to encode this data into a quantum format, commonly referred to as a quantum state (CORTESE; BRAJE, 2018). Consequently, quantum state preparation algorithms (QSPA) emerge as a means of acting as an intermediary in the interaction between classical and quantum computers (SCHULD; PETRUCCIONE, 2018). These algorithms receive a set of classical data

and construct a corresponding quantum state. This is necessary because quantum computers require the initialization of quantum states to perform their functions.

We define a general quantum $|v\rangle$ state as:

Definition 4 *A quantum state $|v\rangle$ can be expressed as a linear combination of elements of an orthonormal basis within a complex Hilbert space H, expressed as:*

$$|v\rangle = \frac{1}{\sqrt{N}} \sum_x c_x |x\rangle ; \quad N = \sum_x |c_x|^2 = 1 \tag{2.18}$$

where $|x\rangle$ is an orthonormal basis for the space, c_x are complex probability amplitudes that determine the probability of measuring the state $|x\rangle$ and N ensures that the total probability sums to one.

Implementing a quantum state refers to the process of transforming a set of qubits, usually in a simple or known state, into a desired quantum state (RATH; DATE, 2024; CORTESE; BRAJE, 2018). That said, we say define:

Definition 5 *An unitary operator SP_v implements a state preparation of a state $|v\rangle$ on n qubits if:*

$$SP_v |0\rangle_n = |v\rangle \tag{2.19}$$

Quantum states can often be grouped into different classes based on their structural properties, such as sparse, dense, entangled, separable, and uniform states, etc. This classification can present opportunities for optimization in quantum algorithms.

In the following sections, we will present examples of state preparation for quantum states.

2.3.1 Merge Algorithm

The state preparation proposed by (GLEINIG; HOEFLER, 2021) describes their contribution as an efficient algorithm designed to take advantage of the sparsity of quantum states. By using this sparsity, the algorithm generates quantum circuits of size $O(|S|n)$, where S denotes the set of basis states with nonzero coefficients, $|S|$ is the amount of nonzero coefficients and n is the number of qubits. The method operates in polynomial classical runtime $O(|S|^2 \log(|S|n))$.

The goal of this method is to construct a quantum circuit applicable to any sparse quantum state (see Definition 4). We describe a general idea of this circuit in Algorithms 1 and 2. This process iteratively combines two sets of basis states in a controlled manner, gradually merging them to construct the desired quantum state. To ensure efficiency, the applied transformations are chosen as general $SU(2)$ rotations, since controlled operations on such matrices have a

linear cost. This reduces the overall complexity of the algorithm while maintaining accuracy in state preparation.

Along with the following algorithms, you can also follow a detailed step-by-step example in APPENDIX A.

Algorithm 1

> **Input:** Classical representation of a quantum state $|v\rangle = \sum_{x \in S} c_x |x\rangle$.
> **Output:** Quantum circuit C that prepares the desired superposition.

1. Start with the initial circuit $C = |0\rangle^{\otimes n}$ and define the set $S = \{x_1, x_2, \cdots x_k\}$ containing the nonzero amplitudes of the target state.

2. Divide the set S into two subsets by selecting a qubit $b \in \{1, 2, \cdots, n\}$ in such a way that:

$$S_0 = \{x \in S \mid x[b] = 0\}; \quad S_1 = \{x \in S \mid x[b] = 1\}$$

   Choose S_0 and S_1 such that the difference in the number of basis states between the two sets is maximized, ensuring that neither set is empty.

3. Select a basis state from each set S_0 and S_1, choosing the pair with the fewest differing qubits, i.e., the most similar states. Then, calculate the angle ω of these states and the relative phase α between them. Let's say we want to merge the arbitrary state $|\phi\rangle$:

$$|\phi\rangle = a |x\rangle + be^{i\theta} |y\rangle$$

   Then,

$$\omega = arctan(\frac{a}{b})$$

   And α is the relative phase between states $|x\rangle$ and $|y\rangle$. If a and $be^{i\theta}$ are the amplitudes:

$$\alpha = Arg(be^{i\theta}) - Arg(a) = \theta$$

4. Apply CNOT gates to the circuit C where the control qubit is b and the target qubits are those that differ between the two states. To merge the set S_0 into S_1, select an open control. To merge the set S_1 into S_0, select a closed control.

5. In the circuit C apply an M gate to the qubit b to "merge" the amplitudes of the two most similar states in S_0 and S_1. This operation is controlled by the other qubits selected for the merge, which should all have the same value at this stage, determining whether the controls are open or closed. M is defined as:

$$M = \begin{bmatrix} \sin(\omega) & e^{i\alpha}\cos(\omega) \\ e^{-i\alpha}\cos(\omega) & -\sin(\omega) \end{bmatrix}$$

6. Replace S with the resulting set after merging.

7. Return the circuit C.

Figure 7 provides a visual representation of Algorithm 2 and its functionality. As shown, after identifying two suitable basis states, the algorithm merges them to form a new state. This results in the system having one fewer basis state, enabling the process to repeat, as illustrated in Algorithm 2.

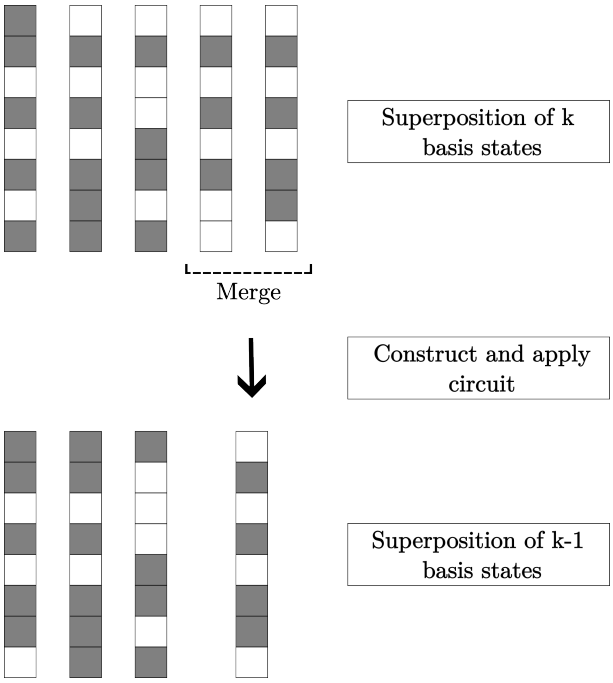


Figure 7 – Illustration of the first step of Algorithm 2, where two suitable basis states are identified and merged to form a new state, reducing the system by one basis state and allowing the process to repeat.

Algorithm 2

Input: Classical representation of a quantum state $|v\rangle = \sum_{x \in S} c_x |x\rangle$.
Output: Quantum circuit C that prepares the desired state preparation.

1. Start with an initial empty quantum circuit C.

2. While the set S contains more than one state, use Algorithm 1 to find a circuit C'.

3. Update the state as: $|v\rangle = C' |v\rangle$

4. Update the quantum circuit as: $C = C' \cdot C$

5. Once S contains only one state $|x\rangle$, apply NOT gates to transform $|x\rangle \rightarrow |0\rangle^{\otimes n}$ and update the circuit.

6. Invert the circuit C to construct a circuit that prepares the original target state $|v\rangle$ from $|0\rangle^{\otimes n}$.

7. Return the final circuit C.

In summary, Algorithm 1 iteratively reduces the set of basis states by merging pairs of states one at a time until only a single state remains. The algorithm constructs a quantum circuit C to prepare a target state by iteratively merging basis states. It begins with $C = |0\rangle^{\otimes n}$ and defines S, the set of computational basis states with nonzero amplitudes. The set is repeatedly partitioned using a chosen qubit b that maximizes the imbalance between subsets S_0 and S_1, ensuring that neither set is empty. The algorithm then selects the most similar states from each subset and computes the merging parameters: the angle ω and the relative phase α.

To align the states for merging, CNOT gates are applied with b as control qubit and the other target qubits corresponding to the different qubits of the states. Then, an M gate – a controlled $SU(2)$ rotation – is applied to b, merging the selected states while preserving their amplitudes and phases. Algorithm 2 builds on this process by applying Algorithm 1 repeatedly to construct a quantum circuit that prepares the desired sparse state. The final steps involve transforming the remaining state into the $|0\rangle^{\otimes n}$ state and then inverting the constructed circuit to reverse the operations, ultimately constructing a circuit that prepares the target state from the initial $|0\rangle^{\otimes n}$.

2.3.2 CVO-QRAM Algorithm

The state preparation method proposed by (VERAS; SILVA; SILVA, 2022) introduces an efficient algorithm that leverages the sparsity of quantum states. It does so by taking into account the characteristics of a double sparse state, that is, states that are sparse in the number of nonzero probability amplitudes (as described in Definition 3), and are also sparse in the number of 1's in the binary representation of the basis states. To illustrate, consider an arbitrary 3-qubit system $|v\rangle$. A double sparse quantum state might look like this:

$$|v\rangle = \frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|010\rangle$$

In this example, the total number of possible basis states is $2^n = 2^3 = 8$. Out of these, only 2 have nonzero amplitudes ($|000\rangle$ and $|010\rangle$). Also, the basis state $|010\rangle$ only has a single 1 in its binary representations, whereas $|000\rangle$ has none. Both this characteristics make this state be considered double sparse.

The associated quantum costs can be seen in the Table 1.

The goal of this method is to construct a quantum circuit applicable to any sparse quantum state, as described in Definition 4. We describe a general idea of this circuit in Algorithm 1. Along with it, you can follow a detailed step-by-step example in APPENDIX B.

Algorithm 1

	CNOT Counts
Sparse States	$\sum_{t=1}^{n} \mu_t(8t-4) - t_{max}$
Dense States	$\sum_{t=1}^{n} C(n,t)(8t-4) - n$

Table 1 – CNOT and Depth Counts for Sparse and Dense States using CVO-QRAM State Preparation. Read μ_t as the number of input patterns c_x with t bits with value 1 in the binary string, t_{max} as the highest value of t and $C(n,t)$ as the binomial coefficient of n and t.

Input: Data consisting of S pairs $data = \sum_{x \in S}\{(x, c_x)\}$
Output: Quantum state of the form $|v\rangle = \sum_{x \in S} c_x |x\rangle$

1. Start with the initial state $|v_0\rangle = |u\rangle |m\rangle$, where:
   - $|u\rangle$ is an ancilla qubit initialized to $|1\rangle$.
   - $|m\rangle$ is a memory register initialized to $|0\rangle^{\otimes n}$, where n is the number of qubits in the system.
2. Sort the input patterns x in ascending order based on the number of bits with value 1. This ordering reduces the number of controlled operations required when processing sparse patterns. The sorted data should be stored in a variable data with the form:
$$data = \sum_{x \in S}\{(x, c_x)\}$$
where c_x is the coefficient associated with each x (see Definition 2.5).
3. For each pair (x, c_x), perform the following steps:
   a) Calculate t, the number of bits with value 1 in x, and identify the list l of the positions of the active bits.
   b) To compute the basis states, apply CNOT operations with the auxilary qubit $|u\rangle$ as the control, targeting the qubits $|m[l_i]\rangle$, where $l_i \in l$, that is:
$$|v_1\rangle = \Pi_{l_i \in l} CX_{(u, m[l_i])} |v_0\rangle$$
   c) Compute the operator $U(x, \gamma)$ controlled by the qubits $m[l]$ and targeted at the qubit $|u\rangle$.
$$U(x,\gamma) = \begin{bmatrix} \sqrt{\frac{\gamma - |x|^2}{\gamma}} & \frac{x}{\sqrt{\gamma}} \\ \frac{-x*}{\sqrt{\gamma}} & \sqrt{\frac{\gamma - |x|^2}{\gamma}} \end{bmatrix}$$
   Calculate γ based on the coefficients that have been processed:
$$\gamma_k = \gamma_{k-1} - |x_{k-1}|^2$$
   By definition, $\gamma_0 = 1$

d) If it is not the final iteration, revert the previously applied CNOT gates.

4. After processing all the pairs (x, c_x), the final state will be:

$$|v\rangle = \sum_{x \in S} c_x |0\rangle |x\rangle$$

The ancilla qubit $|u\rangle$ will be in $|0\rangle$ and can be discarded, resulting in the desired state.

After all patterns are processed, the memory register $|m\rangle$ contains the quantum state desired and the ancilla qubit $|u\rangle$ will be in state $|0\rangle$, indicating that the computation is complete.

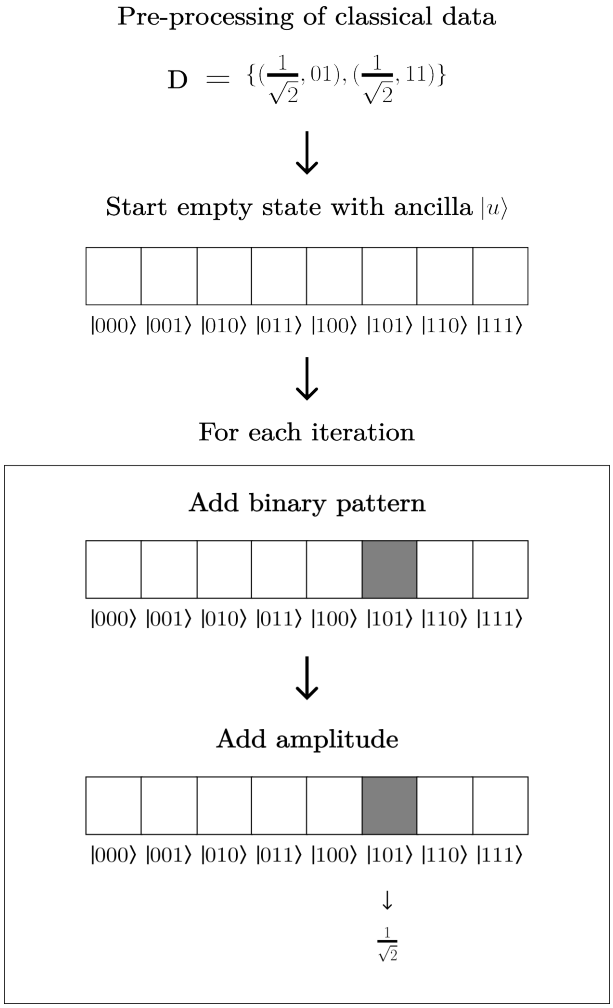


Figure 8 – Illustration of Algorithm 1. The first step is the pre-processing of classical data D. After that, the circuit is started as an empty state with one ancilla. After that, each iteration of the process is responsible for initializing the binary pattern and its corresponding amplitudes, progressing from the simplest to the most complex, enabling system initialization.

Figure 8 illustrates Algorithm 1. After pre-processing the classical data, the algorithm sequentially incorporates binary patterns and their corresponding amplitudes into the circuit, progressing from the simplest to the most complex in each iteration. This process enables the initialization of the entire system.

2.3.3 Pivot Algorithm

The state preparation method introduced by (MALVETTI; ITEN; COLBECK, 2021) offers a strategy for preparing sparse quantum states. It can be implemented using a two-step approach: the first step uses a Pivoting Algorithm, a permutation-based technique that rearranges the amplitudes of the quantum state so that all nonzero entries are clustered together in a specific block of the computational basis. The second step involves applying a dense state preparation algorithm to this nonzero block, reducing the number of qubits involved and consequently minimizing the number of required quantum gates.

By grouping all nonzero entries in a single block and then applying a decomposition used for dense states on the grouped entries, the authors achieve a CNOT count of $(n + 16s - 9)nnz(v) + \frac{23}{24}2^s$, where n is the number of qubits of the state v, $nnz(v)$ is the number of nonzero entries and $s = \lceil \log_2 nnz(v) \rceil$. This method requires no ancilla or at max, one dirty ancilla.

The first step is a pivoting procedure that reorganizes the nonzero amplitudes of the state $|v\rangle$ so that they are concentrated within a single block of computational basis states, referred to as the target block T. To achieve this, the n qubits are conceptually divided into two groups:

- The first $n - s$ qubits index the target block T.
- The remaining $s = \lceil \log_2(nnz(v)) \rceil$ qubits are used to encode the positions of the nonzero amplitudes within the target block T.

This reorganization is performed by a unitary transformation denoted Piv_v, which consists of a sequence of controlled permutations (e.g., multi-controlled NOT gates) that swap zero and nonzero entries until all nonzero amplitudes lie within a block indexed by some $|i\rangle^{\otimes n-s}$. After applying Piv_v, the state is transformed as follows:

$$Piv_v |v\rangle = |i\rangle^{\otimes n-s} \otimes |v'\rangle^{\otimes s} \tag{2.20}$$

where $|v'\rangle^{\otimes s}$ is a normalized quantum state defined on the remaining s qubits.

This pivoting function can be achieved through the application of Algorithm 1, as described by the original work (MALVETTI; ITEN; COLBECK, 2021). Along with it, you can follow a detailed step-by-step example in APPENDIX C.

Algorithm 1

Input: List representation of a quantum state $|v\rangle = \sum c_x |x\rangle$.
Output: Quantum circuit Piv_v that reorganizes the nonzero amplitudes.

1. Let $|v\rangle$ be a quantum state composed of n qubits. If all the nonzero components of $|v\rangle$ are grouped within block T, stop the algorithm.

2. Select a component outside block T with a nonzero amplitude and rewrite it in the form $|t'\rangle^{\otimes n-s}|r'\rangle^{\otimes s}$. Select an entry inside block T with a zero amplitude and rewrite it in the form $|t\rangle^{\otimes n-s}|r\rangle^{\otimes s}$.

3. Choose a qubit where $|t'\rangle^{\otimes n-s}$ and $|t\rangle^{\otimes n-s}$ differ.

4. Using $|t'\rangle^{\otimes n-s}$ as the control qubit, apply at most $n-1$ CNOT gates to adjust $|t'\rangle^{\otimes n-s}|r'\rangle^{\otimes s}$ to a new state $|t''\rangle^{\otimes n-s}|r\rangle^{\otimes s}$, ensuring that t'' and t differ only in the control qubit.

5. Apply an s-controlled NOT gate (controlled by $|r\rangle^{\otimes s}$) to transform $|t''\rangle^{\otimes n-s}|r\rangle^{\otimes s}$ into $|t\rangle^{\otimes n-s}|r\rangle^{\otimes s}$. Note that no other entries in block T are affected during this process.

6. Return to step 1.

Figure 9 provides an illustrative representation of Algorithm 1, showing the initial state $|v\rangle$ and the final state $|u\rangle$. The qubits marked with the block T represent the target block, while the components highlighted in gray have amplitudes of $\frac{1}{\sqrt{2}}$. All other components have zero amplitude.

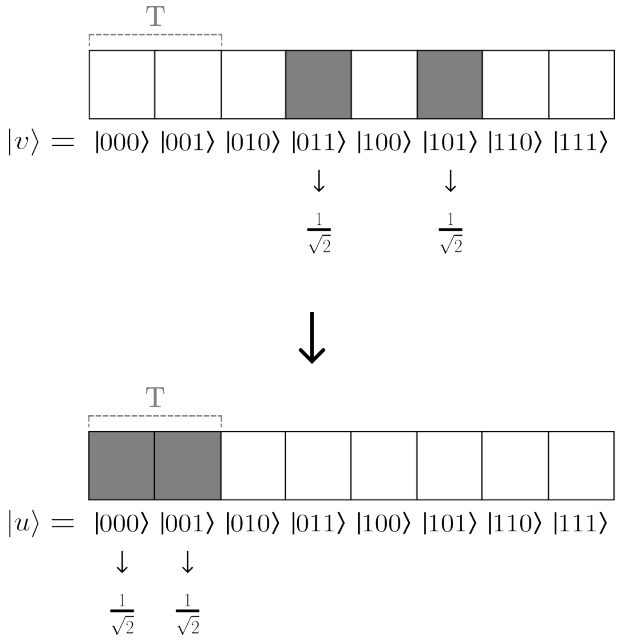


Figure 9 – Illustration of Algorithm 1, showing the initial state $|v\rangle$ and the final state $|u\rangle$. The block T highlights the target qubits, while components in purple have amplitudes of $\frac{1}{\sqrt{2}}$. All other components have zero amplitude.

After the permutation steps, the relative phases of the amplitudes may no longer match those of the original state $|v\rangle$. To address this, a diagonal gate Δ is applied after the pivoting procedure to correct any relative phase errors introduced. It acts only on the s qubits, controlled by the $n-s$ qubits that index the target block T. This gate performs the operation:

$$\begin{aligned}\Delta Piv_v\,|v\rangle =& \Delta(|i\rangle^{\otimes n-s} \otimes |v'\rangle^{\otimes s}) \\ =& (I^{\otimes n-s} \otimes \Delta)(|i\rangle^{\otimes n-s} \otimes |v'\rangle^{\otimes s}) \\ =& |i\rangle^{\otimes n-s} \otimes |\tilde{v}\rangle^{\otimes s}\,,\end{aligned}$$

where $|\tilde{v}\rangle^{\otimes s}$ is a phase corrected version of $|v'\rangle^{\otimes s}$.

Now, the final step consists of applying a state preparation algorithm SP_v on the s qubits that hold the state $|\tilde{v}\rangle$ to map it to $|0\rangle^{\otimes s}$. This gives us:

$$(I^{\otimes n-s} \otimes SP_v^\dagger)(|i\rangle^{\otimes n-s} \otimes |\tilde{v}\rangle) = |i\rangle^{\otimes n-s} \otimes |0\rangle^{\otimes s}$$

Thus, Sparse State Preparation SSP_v can be implemented inverting the operations we derived before:

$$SSP_v = (\Delta Piv_v)^\dagger (I^{\otimes n-s} \otimes SP_{\tilde{v}})$$

The state preparation unitary SP_v implemented in this work employs the Low Rank algorithm (Section 2.3.4). While the Column-by-column approach (ITEN et al., 2016) represents a standard technique within Low Rank state preparation, it was not directly adopted in this work.

2.3.4 Low-Rank Algorithm

The state preparation method proposed by (ARAUJO et al., 2023) reduces the circuit depth of the algorithm by offloading some computational tasks to a classical computer. This method is connected to the Schmidt decomposition, utilizing the Schmidt coefficients as a measure of the entanglement of a quantum state.

Given a quantum state $|v\rangle$ defined on a Hilbert Space $H_A \otimes H_B$, where H_A has n_A qubits and H_B has n_B qubits (the whole system has $n = n_A + n_B$ qubits), its Schmidt decomposition can be expressed as:

$$|v\rangle = \sum_{i=1}^{k} \sigma_i\,|i_A\rangle\,|i_B\rangle \tag{2.21}$$

where:

- k is the Schmidt rank, which characterizes the degree of entanglement between the subsystems A and B;
- σ_i are the Schmidt coefficients, which are non-negative real numbers that satisfy the normalization condition;
- $|i_A\rangle$ and $|i_B\rangle$ are orthonormal bases for the subsystems A and B; and

- $1 \leq i \leq min(dim(H_A), dim(H_B))$.

The authors achieve a computational cost of $O(2^{m+n_B})$ CNOT counts, where $m = \lceil \log_2(k) \rceil < n_A$. If $m = n_A$, this cost becomes $O(2^n)$ CNOT counts. This method doesn't require any ancilla.

The general idea of this algorithm is to take advantage of the entanglement structure of the target state. The algorithm is described in the following. Along with it, you can follow a detailed step-by-step example in APPENDIX D.

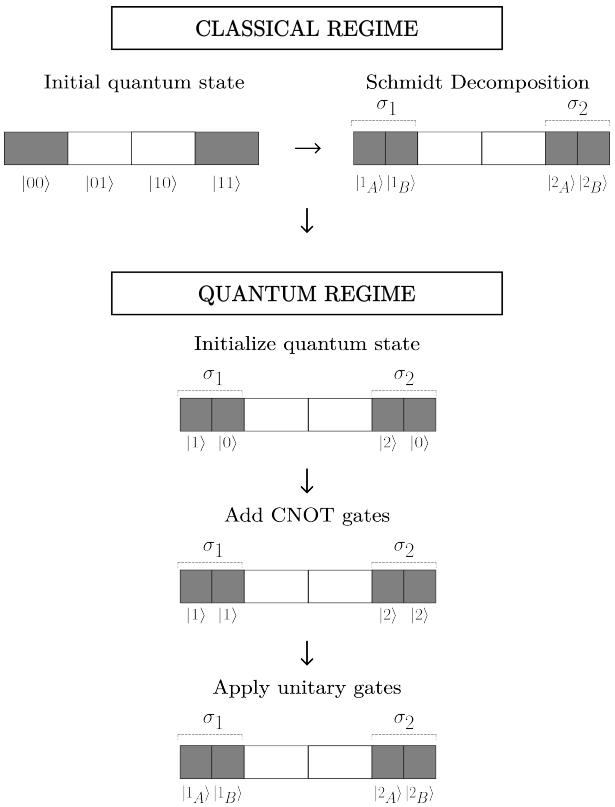


Figure 10 – Illustration of the Low-Rank Algorithm flow. The classical stage performs the Schmidt decomposition. The quantum state initializes a quantum state in a specific form, followed by the addition of CNOT gates. Finally, unitary gates are applied to generate the desired state.

Algorithm 1

1. Compute the Schmidt coefficients σ_i and the corresponding basis states $|i_A\rangle$ and $|i_B\rangle$ by performing the Schmidt decomposition on a classical computer.

2. Initialize a quantum state in the first register enconding the Schmidt coefficients as:

$$\sum_i \sigma_i |i\rangle |0\rangle$$

3. Apply $\lfloor n/2 \rfloor$ CNOT operations to generate entanglement:

$$\sum_i \sigma_i |i\rangle |i\rangle$$

4. Use unitary operations U to the first register and V^T to the second register to map computational bases to Schmidt bases:

$$U|i\rangle = |i_A\rangle\,, \qquad V^T|i\rangle = |i_B\rangle$$

For states with low Schmidt rank, where the Schmidt measure $m = \lceil \log_2(k) \rceil < \lfloor n_A \rfloor$ (where $1 \leq n_a \leq n$), the Low-Rank algorithm substitutes full unitary operations with smaller isometries: $2^m \times 2^{n_A}$ in place of unitary U and $2^m \times 2^{n_B}$ in place of V^T.

Figure 10 illustrates the basic flow of the Low-Rank Algorithm. In the classical stage, the Schmidt decomposition is performed on the input state vector. In the quantum stage, the singular values are initialized as a quantum state, followed by the addition of CNOT gates, and finally, unitary gates are applied to generate the desired state.

2.4 ISOMETRY

An isometry refers to a transformation that preserves distance between metric spaces (WEISSTEIN, n.d.). Given f the function that maps the transformation and d the distance between x and y, elements of two different metric spaces, we have:

$$d(f(x), f(y)) = d(x, y)$$

Reflection, rotation and translation are particular cases of an isometry. A particular example of this is given two pairs of corresponding points (P, P') and (Q, Q'), we have $PQ = P'Q'$; i.e., PQ and $P'Q'$ are congruent segments (have the same shape and size). Also, two figures are said to be congruent if they can be transformed into one another through an isometry (COXETER, 1969).

Definition 6 *An isometry from m to n qubits (where $m \leq n$ and $n \geq 2$) can be represented as a $2^n \times 2^m$ complex matrix V that satisfies:*

$$V^\dagger V = I_{2^n \times 2^n} \tag{2.22}$$

where $V^\dagger$ is the conjugate transpose and $I_{2^n \times 2^n}$ is the $2^n \times 2^n$ identity matrix. Unlike unitary operations, $VV^\dagger$ is not possible unless $m = n$. When $m = n$, the isometry V becomes an unitary matrix, preserving both dimensionality and inner product.

From a pactical perspective, consider a quantum circuit where one input qubit remains fixed in a particular state (e.g., $|0\rangle$), as illustrated in Figure 11. While the circuit's full evolution is described by a unitary matrix, the fixed input reduces the effective degrees of freedom. That is, we can "remove" some of the columns of the matrix without altering the functionality of the circuit. The altered matrix is called isometry.

When we remove some of the columns of the matrix, we reduce the dimensionality without affecting the functionality of the circuit. This allows us to simplify and optimize the circuit representation without sacrificing its properties.

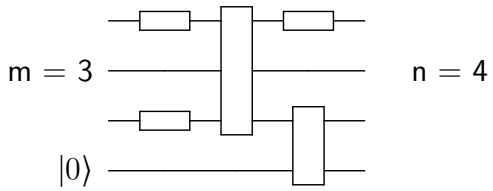


Figure 11 – Example of an isometry in a quantum circuit.

In regards to its content, an isometry can be classified as dense or sparse:

- Similar to dense basis states (see definition 2), we call an isometry dense if it has no (or very few) zero entries, meaning most matrix elements are nonzero. They usually require more quantum gates to implement due to the lack of structure that could be exploited for simplification.
- Like sparse basis states (see definition 2), we call an isometry sparse if it has mostly zero entries. Sparse isometries can often be implemented more efficiently using fewer quantum gates, by taking advantage of their structure.

That said, in this work we focus on sparse isometry decompositions.

2.5 DECOMPOSITION OF SPARSE ISOMETRIES

Among the broad class of isometries, **sparse isometries** tend to stand out because of their potential for optimization (MALVETTI; ITEN; COLBECK, 2021; VERAS et al., 2021; GLEINIG; HOEFLER, 2021). This structure arises in real-world applications, such as simulation of physical open systems (ZHANG et al., 2023; HU et al., 2022), and data encoding for applications in machine learning (LLOYD; MOHSENI; REBENTROST, 2013; REBENTROST; MOHSENI; LLOYD, 2014).

The motivation for studying the initialization of sparse isometries on quantum computers lies in the constraints of current quantum hardware. Devices in the NISQ (Noisy Intermediate-Scale Quantum) era have limited coherence times and gate fidelities, making resource-efficient initialization schemes essential (PRESKILL, 2018). Leveraging the sparsity of a target isometry can significantly reduce the number of quantum gates required, particularly multi-qubit gates such as CNOTs, which are more error-prone (ITEN et al., 2016).

Understanding how to decompose and initialize sparse isometries efficiently is therefore a key step toward scalable quantum algorithms. That said, we will be exploring a tailored

method for preparing sparse isometries based on Householder reflections in the next section (see Section 2.5.1).

2.5.1 Householder Decomposition

Householder Decompositions are a mathematical tool for transforming matrices into simplified forms (HOUSEHOLDER, 1958). In quantum computing, these decompositions are particularly useful for their ability to enable circuit design, such as those implementing isometries. By iteratively transforming a matrix column by column to a simpler form (e.g., diagonal or identity) using Householder reflections, Householder decompositions preserve the orthogonality and unitary properties required by quantum mechanics, enabling efficient and accurate matrix transformations in quantum algorithms.

In this section, we will explore the principles of Householder decompositions and their generalizations to quantum circuits, as demonstrated in the work by (MALVETTI; ITEN; COLBECK, 2021). A key concept in understanding Householder decompositions is the Householder reflection. To illustrate this, consider the geometric representation shown in Figure 12.

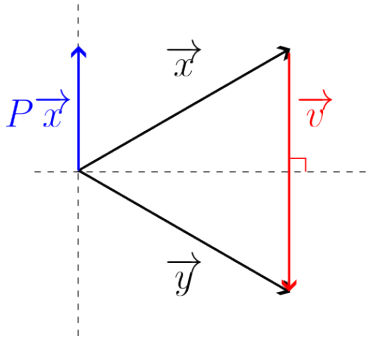


Figure 12 – Geometric representation of the Householder reflection.

We begin by considering two unit vectors, $\vec{x}$ and $\vec{y}$, and assume there exists an operator H that satisfies:

$$H\vec{x} = \vec{y} \tag{2.23}$$

This is the starting point of the deduction, where we seek the operator H that performs a reflection on the vector $\vec{x}$ to map it to $\vec{y}$. From the geometric interpretation of the Householder reflection, as illustrated in Figure 12, we express the vector $\vec{v}$ as the difference between $\vec{x}$ and $\vec{y}$:

$$\vec{v} = \vec{y} - \vec{x} \tag{2.24}$$

Next, we recognize that the projection of $\vec{x}$ onto the space spanned by $\vec{v}$ is related to the reflection process, and we express this projection as:

$$\overrightarrow{v} = -2\mathrm{P}\overrightarrow{x} \tag{2.25}$$

where P is the projection matrix. By isolating the vector $\overrightarrow{y}$ in terms of $\overrightarrow{x}$ from equations 2.24 and 2.25, we obtain the equation:

$$\overrightarrow{y} = (I - 2\mathrm{P})\overrightarrow{x} \tag{2.26}$$

Equation 2.26 demonstrates, when compared to equation 2.23, how the reflection operator H can be expressed as:

$$H = I - 2\mathrm{P} \tag{2.27}$$

Since P represents the projection onto the space spanned by $\overrightarrow{v}$, we express P as $P = |v\rangle\langle v|$, leading to the Householder reflection operator:

$$H_v = I - 2|v\rangle\langle v| \tag{2.28}$$

At this point, we arrive at the definition of the standard Householder reflection. If we introduce a phase ϕ, the reflection operator becomes generalized:

$$H_v^{\phi} = I + (e^{i\phi} - 1)|v\rangle\langle v| \tag{2.29}$$

Thus, we conclude that the equation for the generalized Householder reflection is equivalent to the standard form when $\phi = \pi$, completing the deduction. This process leads us to the formal definition:

Definition 7 *Given an unity vector $|v\rangle$, the standard Householder Reflection with respect to $|v\rangle$ is defined as:*

$$H_v = I - 2|v\rangle\langle v| \tag{2.30}$$

The generalized Householder Reflection of phase ϕ with respect to $|v\rangle$ is defined as:

$$H_v^{\phi} = I - (e^{i\phi} - 1)|v\rangle\langle v| \tag{2.31}$$

The implementation of the Generalized Householder Reflection can take different approaches. For quantum circuits, we can use the state preparation algorithms to execute the Householder Reflections (see Section 5 and Lemma 1).

Lemma 1 *Let SP_v be an unitary operator that performs state preparation for the state $|v\rangle$, and let H_0^{ϕ} be the Householder reflection with respect to $|0\rangle$. Then:*

$$H_v^{\phi} = SP_v \cdot H_0^{\phi} \cdot SP_v^{\dagger} \tag{2.32}$$

Proof 1 *We want to show that:*

$$H_v^\phi = SP_v \cdot H_0^\phi \cdot SP_v^\dagger \quad \text{(I)}$$
$$= I + (e^{i\phi} - 1)\left|v\right\rangle\left\langle v\right| \quad \text{(II)}$$

Let's consider the action of H_v^ϕ acting on a state $|v\rangle$. By definition 5, we know that:

$$SP_v\left|0\right\rangle = \left|v\right\rangle, \quad SP_v^\dagger\left|v\right\rangle = \left|0\right\rangle$$

On the first step shown in equation (I):

$$\begin{aligned} H_v^\phi\left|v\right\rangle =& (SP_v \cdot H_0^\phi \cdot SP_v^\dagger)\left|v\right\rangle = \\ & SP_v \cdot H_0^\phi \cdot (SP_v^\dagger\left|v\right\rangle) = \\ & SP_v \cdot (H_0^\phi\left|0\right\rangle) = \\ & SP_v(e^{i\phi}\left|0\right\rangle) = \\ & e^{i\phi}(SP_v\left|0\right\rangle) = e^{i\phi}\left|v\right\rangle \end{aligned}$$

On the second step shown in equation (II):

$$\begin{aligned} H_v^\phi\left|v\right\rangle =& (I + (e^{i\phi} - 1)\left|v\right\rangle\left\langle v\right|)\left|v\right\rangle = \\ & \left|v\right\rangle(e^{i\phi} - 1)\left|v\right\rangle = \\ & \left|v\right\rangle(1 + e^{i\phi} - 1) = e^{i\phi}\left|v\right\rangle \square \end{aligned}$$

Lemma 2 *The operator H_0^ϕ can be implemented on n qubits using $(n-1)$ controlled gates acting on a single qubit. In the particular case where $\phi = \pi$, H_0 can be implemented using the same number of CNOT gates and auxiliary qubits, along with a NOT gate controlled by $(n-1)$ qubits.*

Proof 2 *To demonstrate the validity of Lemma 2, let's consider a simple example.*

The operator Householder reflection H when $\phi = \pi$ applied to qubit $|0\rangle$ is:

$$H\left|0\right\rangle = (I - 2\left|0\right\rangle\left\langle 0\right|)\left|0\right\rangle = \left|0\right\rangle - 2\left|0\right\rangle = -\left|0\right\rangle$$

Applied to qubit $|1\rangle$ is:

$$H\left|1\right\rangle = (I - 2\left|0\right\rangle\left\langle 0\right|)\left|1\right\rangle = \left|1\right\rangle$$

That is, the matrix representation of the operator H with respect to $|0\rangle$, denoted as H_0, is:

$$H_0 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

Note that implementing the operator H_0 is equivalent to implementing the operator $-Z$. Thus, we can achieve this using three operators on a single qubit: H, X and Z (see Section 2.1 for more details):

$$-Z = XZX = X(HXH)X$$

To make the operation controlled:

$$H_0 = C_{n-1}(-Z) = XH \cdot C_{n-1}(X) \cdot HX \quad \square$$

See Figure 13 for a visual representation of this operation on an arbitrary system.

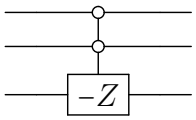


Figure 13 – Controlled $-Z$ operation in a 3-qubit circuit.

The Householder Decomposition process works by receiving as input an isometry V (Section 2.4) of size $2^n \times 2^m$ ($m \leq n$) and has as a goal to reduce V to a simpler form $I_{2^n \times 2^m}$ (first 2^m columns of the identity matrix). If $G = G_k \cdots G_1 G_0 \quad \forall k \in N$ is a product of elementary gates on n qubits such that:

$$GV = I_{2^n \times 2^m} \tag{2.33}$$

Then $G^\dagger$ implements V:

$$\begin{aligned} GV &= I_{2^n \times 2^m} \\ G^\dagger GV &= G^\dagger I_{2^n \times 2^m} \quad \text{(since } G \text{ is unitary, } G^\dagger G = I\text{)} \\ V &= G^\dagger I_{2^n \times 2^m} \quad \square \end{aligned}$$

The first step is to reduce the first column of V denoted $|v_0\rangle$ by applying a Householder Reflection H_{v_0} to map $|v_0\rangle \to |0\rangle$ up to a phase. Then, update the isometry matrix V by applying H_{v_0} to it, which reduces the first column and modifies the other columns. the process involves iteratively applying a sequence of Householder reflections until the isometry is fully reduced to a diagonal form.

Figure 14 illustrates the Householder Decomposition process. It begins with an isometry V composed by zeros and arbitrary complex numbers $*$, where the first step involves applying the Householder reflection H_{v_0}, transforming V into a new isometry. Through a sequence of operations, at most 2^m in total, the isometry is progressively reduced to a diagonal matrix.

$$V = \begin{bmatrix} 0 & 0 & 0 & * \\ * & * & * & * \\ 0 & 0 & 0 & 0 \\ 0 & * & * & 0 \\ 0 & 0 & 0 & 0 \\ * & 0 & 0 & * \\ 0 & 0 & 0 & 0 \\ 0 & * & 0 & 0 \end{bmatrix} \xrightarrow{H_{v_0}} H_{v_0}V = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & * & 0 \\ 0 & * & 0 & * \\ 0 & * & 0 & 0 \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & * & 0 & * \end{bmatrix} \xrightarrow{\cdots} (H_{v_3}H_{v_2}H_{v_1}H_{v_0})V = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 14 – Illustration of the Householder decomposition. The process begins with an isometry V, composed by zeros and arbitrary complex numbers $*$, where the first step involves applying the Householder reflection H_{v_0}, transforming V into a new isometry. After at most m operations, the isometry is reduced to a diagonal matrix.

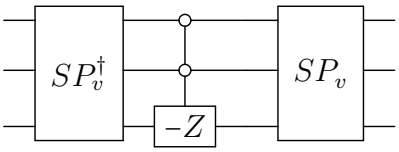


Figure 15 – Quantum circuit illustrating the first step of a possible Householder decomposition. In this step, a state preparation is applied, followed by a controlled $-Z$ gate acting on the third qubit, and concluding with the application of the transposed conjugate of the state preparation.

Figure 15 illustrates the first step of a potential Householder decomposition in a quantum circuit. The process starts with the application of the conjugate transpose of the state preparation, followed by the application of a controlled $-Z$ gate, which represents one possible implementation of the Householder reflection (see the Proof for Lemma 2), on the third qubit. The step concludes with the application of the state preparation, completing the operation.

3 METHODOLOGY

The main goal of this work is to analyze and compare different state preparation methods for decomposing sparse isometries. To do that, we explore three approaches based on Householder decomposition: Pivot (see subsection 2.3.3), Low Rank (see subsection 2.3.4), and Merge (see subsection 2.3.1). We focus on evaluating how these methods perform, especially in terms of efficiency and suitability for quantum devices with limited resources.

The original work by Malvetti et al. (MALVETTI; ITEN; COLBECK, 2021) employs the Pivot-based state preparation as the core strategy for isometry decomposition. In this study, we extend that analysis by applying the Merge and Low Rank methods in place of Pivot and comparing their results. This allows us to investigate how different state preparation strategies impact the overall resource cost and circuit structure of the final isometry implementation.

We also compare our results with existing tools available online, particularly the Column-by-column Decomposition (ITEN et al., 2016) implemented by Qiskit—currently the most widely used quantum computing library for Python. The idea is to see how our methods stack up against what is already out there.

We use the CNOT count and circuit Depth as key metrics to assess the impact of state preparation on the Householder decomposition. These metrics provide insights into the efficiency and complexity of implementing the decomposition in quantum circuits.

To evaluate the effect of state preparation, we consider different values of s, a sparsity parameter that grows with the density of the isometry, where $s = \{1, 2, 3, 4\}$, across a range of number of qubits from 2 to 12 (see section 3.2). This allows us to analyze how the choice of s influences the structure and performance of the quantum circuit. Additionally, the analysis spans isometries with dimensions corresponding to $m = \{1, 2, 3, 4\}$ (see Definition 6, where 2^m is the number of columns of the isometry), enabling a comprehensive exploration of how varying matrix sizes affect the overall complexity of the decomposition process.

This comparison helps highlight the strengths and weaknesses of each approach and provides useful insights into improving state preparation techniques, especially for cases involving sparse states and specific structural patterns.

In this chapter, we take a closer look at how the algorithm works and how it can be implemented. We will go through the main ideas behind it and provide some practical insights into its design and execution.

3.1 HOUSEHOLDER DECOMPOSITION ALGORITHM

The Householder Decomposition Algorithm implements isometries in quantum circuits by breaking them down into simpler operations using Householder reflections. These reflections map columns of the matrix to basis vectors, ensuring orthogonality and preserving unitary

nature throughout the process (see section 2.5).

To implement this algorithm, we divide the process into three key steps: the Householder Reflection with respect to the state $|0\rangle$ (see Lemma 2), the Householder Reflection with respect to the state $|v\rangle$ (see Lemma 1), and the Householder Decomposition.

3.1.1 Implementation of Householder Reflection with Respect to the State $|0\rangle$

The algorithm for implementing the Householder Reflection with respect to the state $|0\rangle$ or H_0^ϕ (see Lemma 2), described in Algorithm1, constructs a quantum circuit that performs this reflection. It is a subcircuit (gate) used to implement the Householder Reflection with respect to $|v\rangle$. It achieves this by defining a unitary operation U and applying its multi-controlled version within the circuit. The resulting quantum circuit effectively realizes the Householder reflection as a multi-controlled unitary gate.

Algoritmo 1: H_0^ϕ or Householder Reflection with respect to $|0\rangle$

Input: Number of qubits n and angle ϕ
Output: Multicontrolled unitary gate circuit

```
1  Function Householder Reflection( n, φ):
2      Initialize quantum circuit:
3      circuit ← |0⟩^⊗n
4      Initialize unitary operation:
5      U = [ e^{iφ}  0 ]
           [ 0       1 ]
6      Create controlled operation:
7      MCU ← C_{n-1}(U)
8      Apply controlled operation to circuit:
9      circuit ← MCU
10     return circuit
11 end
```

The algorithm takes as input two parameters: the number of qubits n and the angle ϕ used in the unitary transformation, which will control the phase in the reflection operation. In line 3, the quantum circuit is initialized with the state $|0\rangle^{\otimes n}$, which is the tensor product of n qubits, all initialized to the state $|0\rangle$. In line 5, a unitary operation U is defined in matrix form and represents a phase shift by ϕ applied to the first component of the state vector. Then, in line 7, a multi-controlled unitary operation MCU is constructed by applying the controlled version of U on the quantum circuit, where $C_{n-1}(U)$ represents applying U to the target qubit, controlled by the $n-1$ control qubits. We chose the controlled operation defined in section 2.2.1. In line 9 the controlled operation MCU is applied to the quantum circuit and, finally, in line 10, the algorithm returns the modified quantum circuit with the Householder Reflection with respect to $|0\rangle$.

The quantum circuit representation of this algorithm is illustrated in Figure 16. It features open controls on the control qubits and the gate U is applied to the target qubit only when all preceding qubits are in the $|0\rangle$ state. It adds a phase ϕ to the $|0\rangle$ components only.

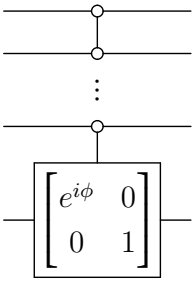


Figure 16 – Controlled U operation in a n-qubit circuit. It is a subcircuit used to implement the Householder Reflection with respect to $|v\rangle$ algorithm (see Section 3.1.2), where a single-qubit phase gate that applies a phase ϕ to the $|0\rangle$ component only.

3.1.2 Implementation of Householder Reflection with Respect to the state $|v\rangle$

The algorithm for implementing the Householder Reflection with respect to the state $|v\rangle$, described in Algorithm 2 constructs a quantum circuit that performs this reflection by leveraging state preparation techniques and controlled unitary operations.

Algoritmo 2: H_v^ϕ or Householder Reflection with respect to $|v\rangle$

Input: Amplitude vector v representing the state $|v\rangle$
Number of qubits n
State preparation SP
Output: Quantum circuit implementing the Householder Reflection

Function `Generalized Householder Reflection`(v, n, SP)**:**
  Initialize quantum circuit:
  $circuit \leftarrow |0\rangle^{\otimes n}$
  Compute angle ϕ for the Householder reflection:
  $\phi \leftarrow$ angle(v)
  Applies the inverse of the state preparation to the circuit:
  $circuit \leftarrow SP^\dagger(v)$ (see Section 2.3)
  Apply Householder reflection with respect to $|0\rangle$ to circuit:
  $circuit \leftarrow$ HouseholderReflectionZero(num_qubits, ϕ) (see algorithm 1)
  Apply state preparation to circuit:
  $circuit \leftarrow SP(v)$
  return $circuit$
end

The algorithm takes as input three parameters: an amplitude vector v, that represents the quantum state $|v\rangle$, which defines the reflection axis. This vector encodes the amplitudes of the state to be reflected; the number of qubits n, needed to represent the quantum state $|v\rangle$ in the computational basis; and a state preparation operator SP, a quantum operation responsible for preparing the state $|v\rangle$ from the initial zero state $|0\rangle^{\otimes n}$. As an output, we have quantum circuit that implements the Householder Reflection about the state $|v\rangle$.

In line 3, the quantum circuit is initialized in the state $|0\rangle^{\otimes n}$, representing n qubits in the zero state. In line 5, the algorithm computes an angle ϕ, which parameterizes the phase shift needed for the Householder reflection. This angle depends on the amplitudes of the input vector v. To enable reflection with respect to $|v\rangle$, the inverse state preparation operator $SP^{\dagger}(v)$ is applied, as shown on line 7. This operation maps $|v\rangle$ back to $|0\rangle^{\otimes n}$, effectively transforming the reflection problem into one with respect to $|0\rangle$. The pre-computed Householder Reflection with respect to $|0\rangle$ is applied to the circuit as inversed (see section 1). This operation introduces the desired phase shift ϕ around the $|0\rangle$ axis, effectively simulating the reflection about $|v\rangle$ in the original space. After applying the reflection, the original state preparation operator $SP(v)$ is re-applied in line 11 to return the quantum state to its original configuration. The resulting circuit implements the desired Householder Reflection with respect to $|v\rangle$.

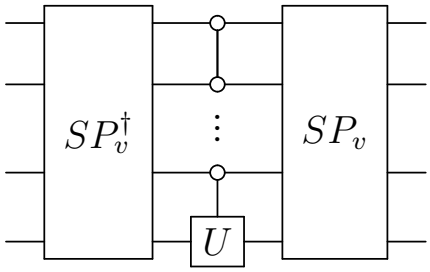


Figure 17 – Quantum circuit of the Generalized Householder Reflection. The inverse of a state preparation is applied, followed by a controlled unitary gate U acting on the target qubit, and concluding with the application of the state preparation.

3.1.3 Implementation of Householder Decomposition

The algorithm for implementing the Householder Decomposition, described in Algorithm 3, constructs a quantum circuit that implements an isometry by performing a sequence of Householder Reflections in the columns of the isometry until the final isometry the sequence of these reflections turns the original isometry to a diagonal matrix.

This algorithm takes as input an isometry W with dimensions $2^n \times 2^m$ and outputs a quantum circuit that implements the isometry. In line 3, we determine the number of qubits n required for the circuit by computing the logarithm of the number of rows in the isometry. Next, in line 5, we initialize a quantum circuit with n qubits, which serves as the foundation for implementing the transformation. In line 7, we define a diagonal matrix, which is simply an identity matrix with the same dimensions as the input isometry. This matrix acts as a reference

point to iteratively transform the input isometry into a diagonal form. The loop starting at line 9 iterates through the columns of the isometry matrix, processing them sequentially. This iterative approach allows us to apply Householder Reflections column by column, progressively simplifying the isometry until it matches the diagonal form. In the conditional statement at line 13, we check whether the current column of the isometry already matches the corresponding column of the diagonal matrix. If they are equal, we skip further processing for that column and proceed to the next iteration. In line 18, we compute the Householder vector by subtracting the current column of the isometry from the corresponding column of the diagonal matrix. This vector is used to construct the reflection that aligns the isometry with the diagonal matrix. In line 20, we normalize the Householder vector. This step is essential because the vector must have unit length to properly define the reflection applied in line 22. In line 24, the Householder reflection is applied to the initialized quantum circuit, implementing the transformation associated with the current column. Subsequently, in line 26, we update the isometry by applying the same reflection, ensuring it progressively aligns with the diagonal matrix as the loop continues. After processing all columns, the final circuit inversed is returned as the output. When applied in a quantum computation, this circuit implements the desired isometry, enabling transformations required for state preparation, data embedding, or other quantum algorithms.

The figure 18 illustrates the quantum circuit for the Householder Decomposition applied to n qubits, processing m columns of the matrix. In the figure, SP_{v_i} represents the state preparation applied at step i, while $SP_{v_i}^\dagger$ denotes its Hermitian adjoint, corresponding to the inverse operation. The unitary gate U_i is a multi-controlled operation associated with the vector $|v_i\rangle$, enabling the transformation required for each step of the decomposition process.

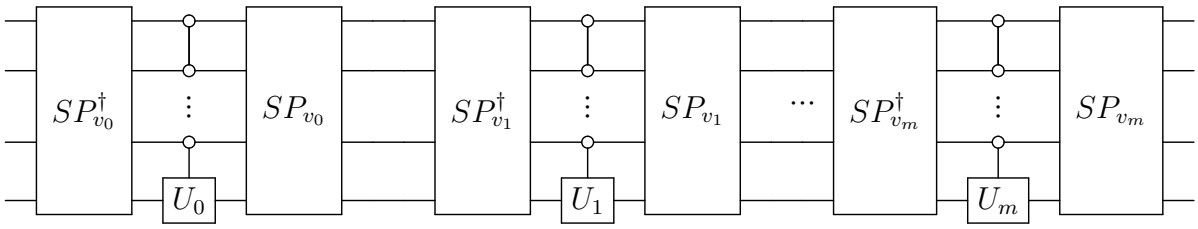


Figure 18 – Illustration of a Householder Decomposition circuit applied to n qubits, processing m columns of the matrix. The operation SP_{v_i} represents the state preparation at step i, while $SP_{v_i}^\dagger$ denotes its Hermitian adjoint (conjugate transpose). The gate U_i corresponds to the multi-controlled unitary operation associated with the vector $|v_i\rangle$.

Algoritmo 3: Householder Decomposition for Isometry

Input: Isometry W $(2^n \times 2^m)$
Output: Quantum circuit implementing W
Function `Householder Decomposition(`W`)`**:**
  Calculate amount of qubits n **in the isometry:**
  $n \leftarrow log_2$(Number of rows of W)
  Initialize quantum circuit:
  $circuit \leftarrow |0\rangle^{\otimes n}$
  Initialize a $2^n \times 2^m$ **diagonal matrix:**
  $D \leftarrow I_{2^n,2^m}$
  $j \leftarrow 0$
  while $(W \neq D)$ *and* $(j < m)$ **do**
    Extract columns:
    $w_j \leftarrow$ Column j of W
    $d_j \leftarrow$ Column j of D
    if $w_j \approx d_j$ **then**
      $j \leftarrow j + 1$
      continue
    end
    Compute Householder vector:
    $v_j \leftarrow d_j - w_j$
    Normalize:
    $v_j \leftarrow \frac{v_j}{\|v_j\|}$
    Compute Householder Reflection:
    $H_j \leftarrow$ HouseholderReflection(v_j) (see Algorithm 2)
    Apply reflection to the circuit:
    $circuit \leftarrow circuit \cdot H_j$
    Update isometry:
    $W \leftarrow H_j \cdot W$
    $j \leftarrow j + 1$
  end
  return $circuit.inverse()$
end

3.2 SPARSE MATRICES

To generate random sparse matrices, we first need to define the number of qubits n of the system, as it defines the Hilbert space size for the quantum system.

For a quantum system with n qubits, the total dimension of the Hilbert space is $2^n \times 2^n$. This size determines the final shape of the output matrix. For example:

- **1 qubit:** $2^1 \times 2^1 = 2 \times 2$
- **2 qubit:** $2^2 \times 2^2 = 4 \times 4$
- **3 qubit:** $2^3 \times 2^3 = 8 \times 8$

The function partitions this large space into smaller blocks of dimension $2^s \times 2^s$, where s is the number of qubits that defines each block. Consequently, the number of blocks x is given by $x = 2^{n-s}$. This means the number of blocks decreases as s increases, leading to larger blocks and lower sparsity.

The parameter s must satify the condition that $s \leq n$, so that we can ensure that the block size $(2^s \times 2^s)$ does not exceed the overall dimension of the the final matrix $(2^n \times 2^n)$.

Given a list of unitary $2^s \times 2^s$ matrix $U = \{U_1, U_2, \cdots, U_x\}$, in the shape:

$$U_i = \begin{pmatrix} u_{11}^{(i)} & u_{12}^{(i)} & \cdots & u_{1,2^s}^{(i)} \\ u_{21}^{(i)} & u_{22}^{(i)} & \cdots & u_{2,2^s}^{(i)} \\ \vdots & \vdots & \ddots & \cdots \\ u_{2^s,1}^{(i)} & u_{2^s,2}^{(i)} & \cdots & u_{2^s,2^s}^{(i)} \end{pmatrix}$$

To construct the full sparse unitary matrix, we place the blocks U_i along the diagonal, resulting in a block diagonal matrix of shape $2^n \times 2^n$ with $x = 2^{n-s}$ blocks, the structure becomes:

$$\begin{pmatrix} U_1 & 0 & \cdots & 0 \\ 0 & U_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & U_x \end{pmatrix}$$

Next, we randomly permute the rows and columns of the matrix. Notice that when $s = 1$, the blocks are the smallest possible unitary matrices (i.e., 2×2), resulting in a sparse structure with many zeros filling the off-diagonal elements, having maximum sparsity. On the other hand, increasing s leads to larger blocks, reducing sparsity and making the matrix more dense. If $s = n$, the matrix becomes entirely dense, containing a single block.

4 EXPERIMENTS AND RESULTS

In this chapter, we present the results of our experiments, which were conducted using numerical simulations of Householder Decompositions (see Section 2.5) applied to random sparse isometries (see Section 3.2). To generate the unitary matrices used, we use the function `unitary_group.rvs` from `scipy.stats.unitary_group`. The distribution of these matrices is not arbitrary but follows the Haar distribution on the unitary group. This means that the matrices are uniformly distributed across the unitary group, ensuring that all possible unitary matrices have an equal probability of being selected. The Haar distribution also does not change under multiplication, meaning that the probability of any particular matrix being chosen is invariant regardless of the matrix's position within the group (MEZZADRI, 2007). As a result, the random isometries generated for our experiments accurately reflect the variety of unitary transformations, offering a solid foundation for evaluating the performance of the decomposition methods.

The primary objective of these experiments is to assess the effectiveness and efficiency of reducing the computational complexity associated with decomposing sparse isometries, aiming to achieve better performance than existing methods. For practical implementation, the state preparation methods used in these experiments are available in the publicly accessible library qclib, which offers a comprehensive set of tools for quantum computing.

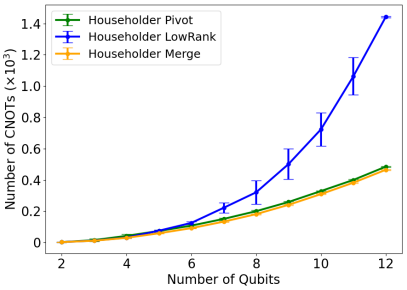
4.1 CNOT ANALYSIS

We begin by analyzing the cases where $m = 1$. For this scenario, we fix the value of m and vary s within the range $1 \leq s \leq 4$.

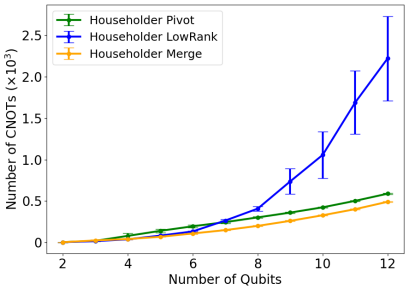
The Merge state preparation (see Subsection 2.3.1) demonstrates the greatest advantage among the methods analyzed. Compared to both Pivot State Preparation (see Subsection 2.3.3) and the Low Rank methods (see Subsection 2.3.4), the Merge State Preparation within the Householder decomposition exhibits superior performance starting from 6 qubits, as illustrated in Figure 19. In particular, the growth rate of the Merge curve shows minimal sensitivity to variations in s when the isometry is small ($m = 1$). Moreover, its error bars are too small to be visible relative to the other state preparation methods.

The Pivot State Preparation exhibits behavior similar to the Merge method when s is small. However, for denser isometries, it displays a noticeable peak in values, particularly within the range of 4 to 9 qubits, as shown in Figures 20c and 20d. In these regions, the error bars are considerably larger, reflecting greater variability and instability. More investigation is needed to fully understand this outcome.

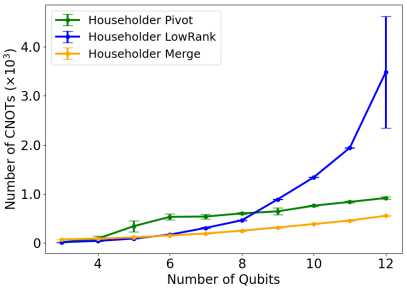
In contrast, the Low Rank State Preparation demonstrates the highest growth rate among the three methods but maintains a relatively consistent behavior overall. An interesting



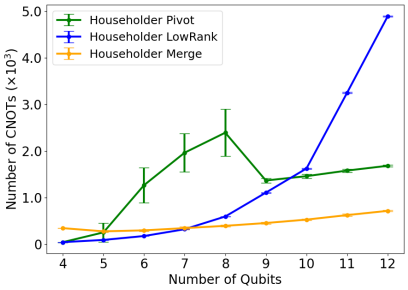
(a) $s = 1,\ m = 1$



(b) $s = 2,\ m = 1$

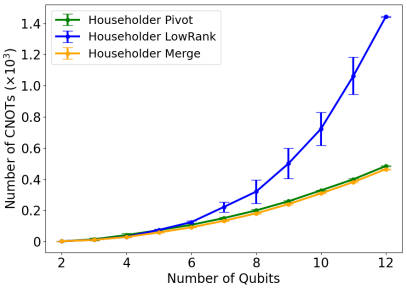


(c) $s = 3,\ m = 1$

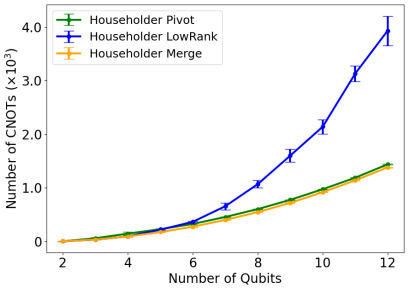


(d) $s = 4,\ m = 1$

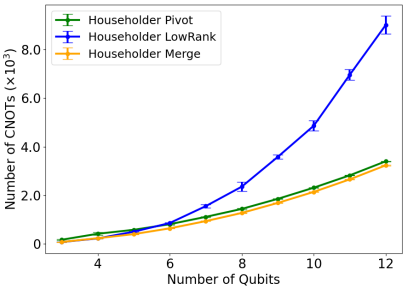
Figure 19 – Number of qubits versus number of CNOT gates for varying $1 \leq s \leq 4$ with $m = 1$ fixed. The curves correspond to different state preparation methods applied to Householder Decompositions—green represents Pivot, blue represents Low Rank, and orange represents Merge .



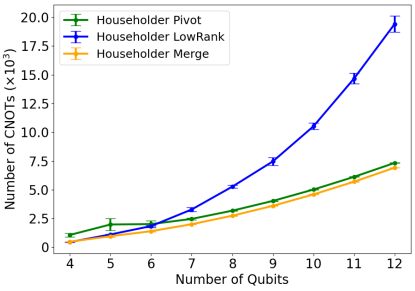
(a) $s = 1,\ m = 1$



(b) $s = 1,\ m = 2$



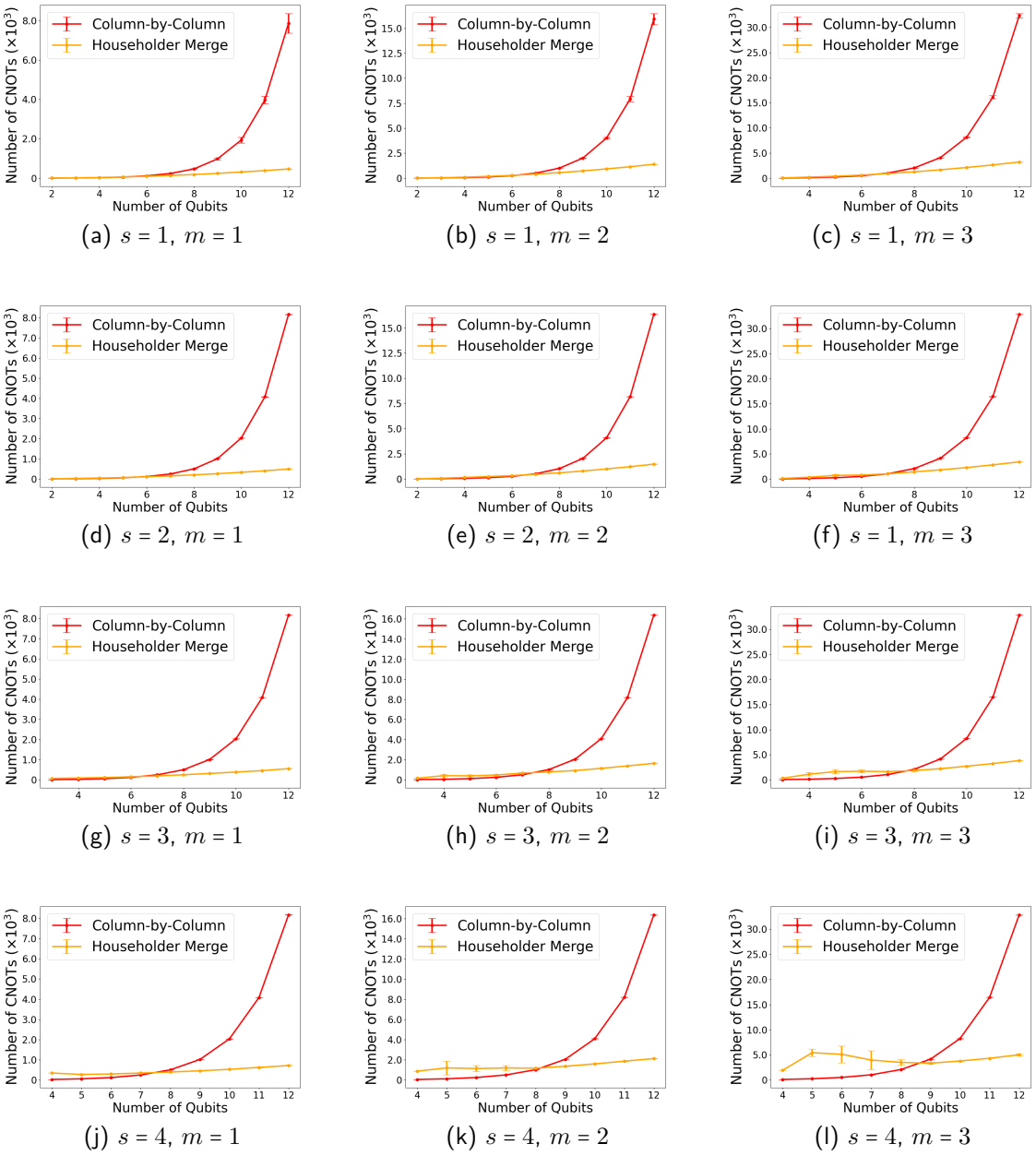
(c) $s = 1,\ m = 3$



(d) $s = 1,\ m = 4$

Figure 20 – Number of qubits versus number of CNOT gates for varying $1 \leq m \leq 4$ with $s = 1$ fixed. The curves correspond to different state preparation methods applied to Householder Decompositions—green represents Pivot, blue represents Low Rank, and orange represents Merge.

observation regarding the Low Rank method is its error bars, which tend to decrease for denser isometries. It should be noted that the Low-Rank method assumes, by default, a dense state. As a result, they naturally produce more complex circuits when compared to approaches optimized for sparse states. This inherent characteristic explains their higher resource overhead, particularly in scenarios where state sparsity could otherwise be exploited. However, an exception occurs in the case where $s = 3$ and $m = 1$ (Figure 20c), where a single data point exhibits a notable large error bar, indicating an anomaly in this particular instance. The anomaly likely arises from states that are exactly separable under the fixed Low-Rank bipartition, making the decomposition less costly and producing outliers.



(a) $s = 1,\ m = 1$ (b) $s = 1,\ m = 2$ (c) $s = 1,\ m = 3$

(d) $s = 2,\ m = 1$ (e) $s = 2,\ m = 2$ (f) $s = 1,\ m = 3$

(g) $s = 3,\ m = 1$ (h) $s = 3,\ m = 2$ (i) $s = 3,\ m = 3$

(j) $s = 4,\ m = 1$ (k) $s = 4,\ m = 2$ (l) $s = 4,\ m = 3$

Figure 21 – Number of Qubits vs Number of Cnots for varying $1 \leq s \leq 4$ and $1 \leq s \leq 3$. The curves correspond to isometry decompositions – red represents the Column-by-column isometry method and orange represents The Householder Merge Decomposition.

Next, by fixing the value of s and varying m within the range $1 \leq m \leq 4$ we observe the behavior shown in Figure 20. Once again, the Merge state preparation demonstrates the lowest number of CNOTs among the three methods. The overall trend of the graphs remains relatively unchanged with varying m. However, the CNOT cost increases as m grows, which is expected, as the isometry becomes larger with increasing m.

Compared to the isometry decomposition method implemented by Qiskit, the Column-by-column method (ITEN et al., 2016), our proposed approach demonstrates clear advantages for highly sparse isometries, starting from 7 qubits. This observation is evident in the experimental cases where $1 \leq s \leq 2$ and $1 \leq m \leq 2$, as well as in the case where $s = 3$ and $m = 1$ (see Figures 21a through 21g). As the isometry becomes denser and the system size increases, our method continues to outperform Qiskit, with advantages appearing at 8 qubits. However, it is important to note that, for denser states, our algorithm exhibits some variability in its performance, particularly for smaller base cases (see Figure 21l). The reason behind this variability is still unclear and requires further investigation.

This performance can be understood by considering that the method used (ITEN et al., 2016) assume dense state preparation. Consequently, they naturally produce more complex circuits, as they are not optimized to exploit sparsity. While these methods provide reliable performance across general cases, their circuit complexity remains higher, particularly when compared to our method's ability to leverage sparsity for more efficient decompositions.

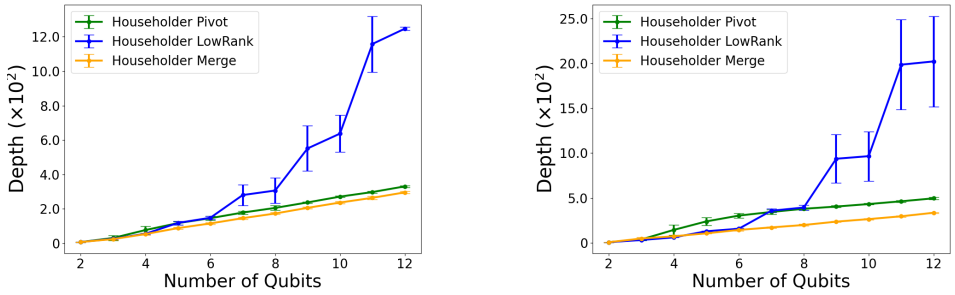
4.2 DEPTH ANALYSIS

Just like in section 4.1, we start by comparing the cases where $m = 1$. We fix the value of m and vary s within the range $1 \leq s \leq 4$. See figure 22.

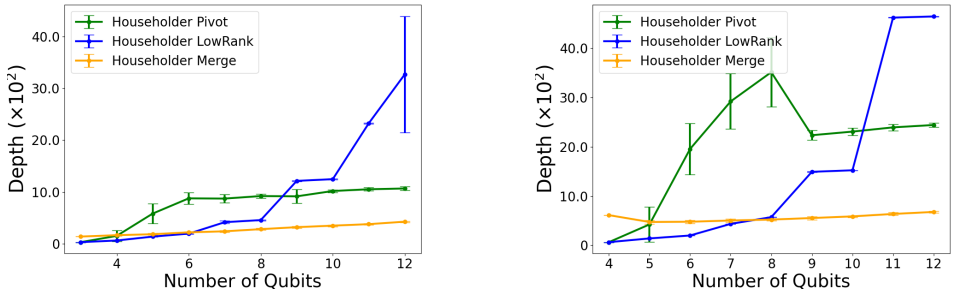
We observe that the Merge State Preparation exhibits the most significant advantages in terms of circuit depth when compared to the Pivot and Low Rank State Preparation methods within the framework of the Householder Decomposition. These advantages become evident for systems with at least 7 qubits, as illustrated in Figures 22a through 22c. It is worth noting that the standard deviation bars are so small that they are not clearly visible in all cases. In addition to achieving the lowest circuit depth, the Merge State Preparation also demonstrates the slowest growth rate relative to the other methods.

In contrast, the Pivot State Preparation displays the most variable behavior among the approaches analyzed. Specifically, it exhibits a pronounced growth peak as the parameter s increases, particularly within the range of $5 \leq n \leq 9$ qubits, as highlighted in Figure 22d. Furthermore, its standard deviation increases as the isometry becomes denser.

The Low Rank State Preparation, while exhibiting the highest growth rate among the three methods, appears more consistent than the Pivot State Preparation. Unlike the Pivot method, the standard deviation bars for the Low Rank approach tend to increase with the number of qubits. This behavior occurs because as the system size grows, more states happen to



(a) $s = 1,\ m = 1$ (b) $s = 2,\ m = 1$



(c) $s = 3,\ m = 1$ (d) $s = 4,\ m = 1$

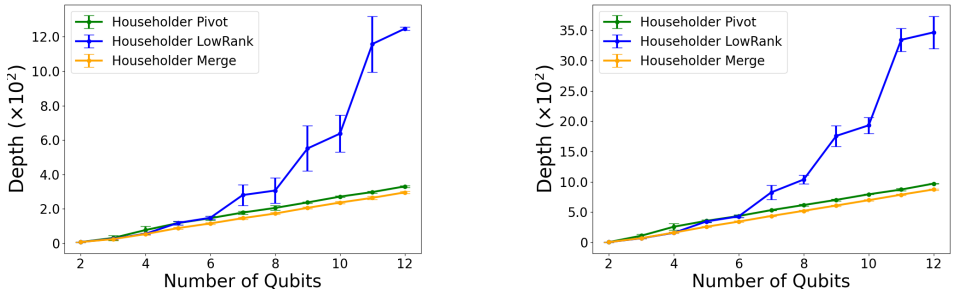
Figure 22 – Number of qubits versus Depth count for varying $1 \leq s \leq 4$ with $m = 1$ fixed. The curves correspond to different state preparation methods applied to Householder Decompositions—green represents Pivot, blue represents Low Rank , and orange represents Merge.

satisfy the separability condition under the default bipartition configuration, leading to greater variance in the decomposition costs. However, an anomaly is observed in Figure 22d, where the experimental results are exceptionally consistent despite the expected discrepancies. This can be explained by considering that with denser states, the probability of encountering a separable state under the default bipartition configuration drops rapidly, leading to greater variance in the decomposition costs.

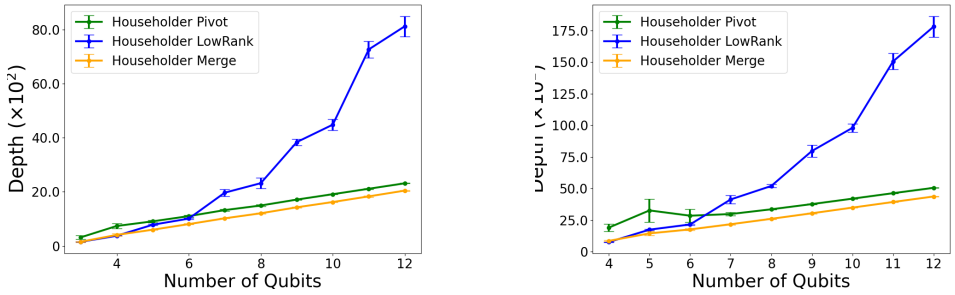
Next, by fixing the value of s and varying m within the range $1 \leq m \leq 4$, we observe the behavior shown in Figure 23. For low sparsity, we observe that the Merge and Pivot State Preparation methods exhibit similar growth patterns, with Merge demonstrating slightly better performance, as seen through Graphs 23a through 23d. Notably, Merge achieves the lowest circuit depth, a trend observable from as early as 7 qubits. In the initial base cases – up to 6 qubits – the depth counts are comparable across all state preparation methods.

As the size of the isometry increases (i.e., larger m), the Pivot method displays a pronounced growth peak in the earlier cases, as seen particularly in Graph 23d.This behavior warrants additional investigation. In contrast, the Low Rank method exhibits the largest overall growth, although its standard deviation decreases for larger isometries. The Pivot method's growth peak is accompanied by the highest standard deviation, whereas the Merge method maintains consistently low standard deviation values throughout.

Compared to the isometry decomposition method available in Qiskit (ITEN et al., 2016),



(a) $s = 1,\ m = 1$ (b) $s = 1,\ m = 2$



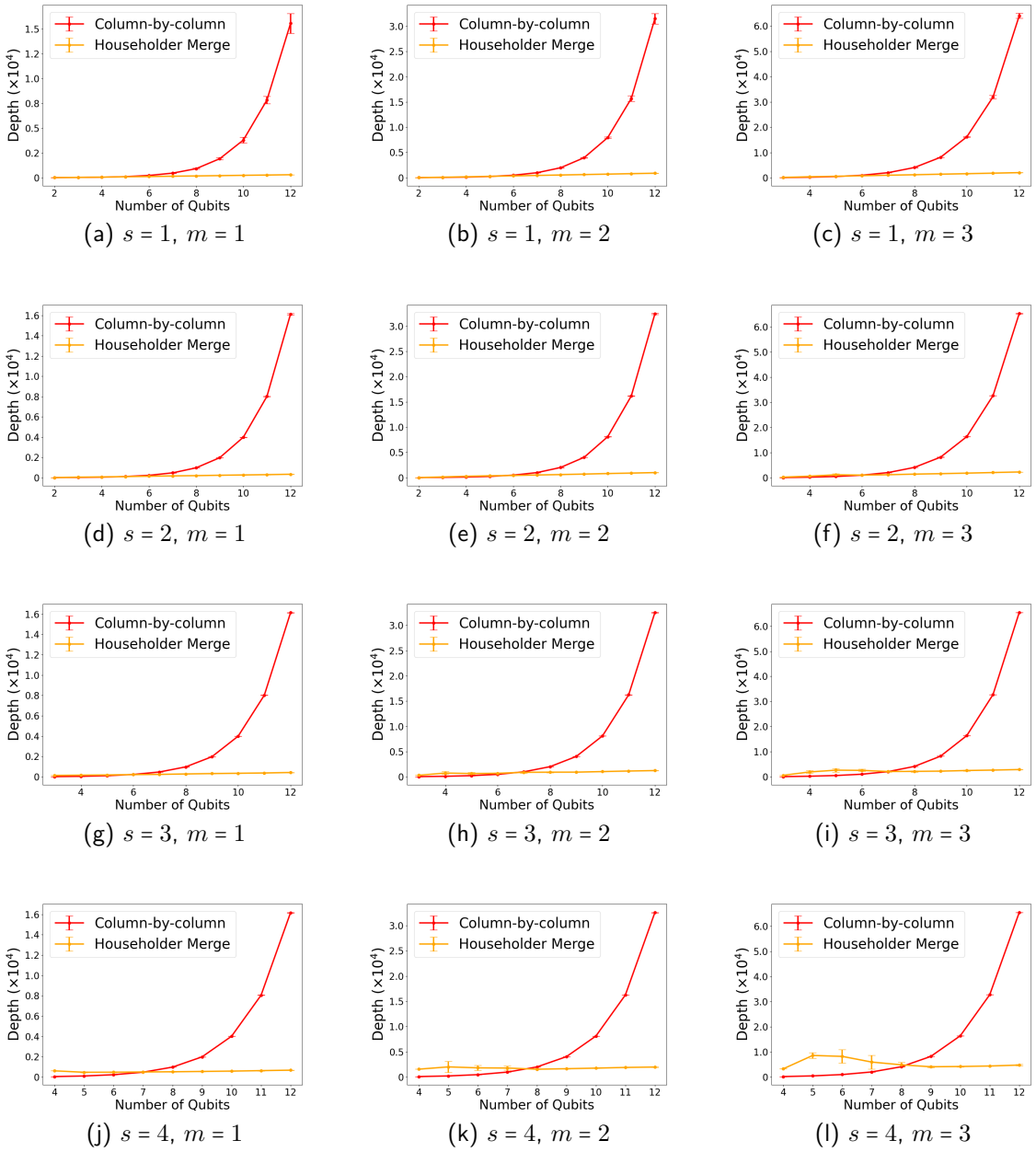
(c) $s = 1,\ m = 3$ (d) $s = 1,\ m = 4$

Figure 23 – Number of qubits versus Depth count for varying $1 \leq m \leq 4$ with $s = 1$ fixed. The curves correspond to different state preparation methods applied to Householder Decompositions—green represents Pivot, blue represents Low Rank, and orange represents Merge.

our approach demonstrates advantages beginning at 7 qubits for the ranges $1 \leq s \leq 2$ and $1 \leq m \leq 3$, as well as for the specific case where $s = 3$ and $m = 1$. These trends are illustrated in Figures 24a through 24f, and Figure 24g.

For denser or larger isometries, these advantages are observed starting at 8 qubits, as shown in Figures 24i through 24l. Additionally, it is noteworthy that the Merge State Preparation exhibits a distinct growth peak as the isometries become denser and larger.

As in the CNOT analysis, the observed performance is likely explained by the fact that Qiskit's method (ITEN et al., 2016) assumes dense state preparation. Consequently, it tends to generate more complex circuits, as it does not exploit the potential sparsity of the target states.



(a) $s = 1,\ m = 1$ (b) $s = 1,\ m = 2$ (c) $s = 1,\ m = 3$

(d) $s = 2,\ m = 1$ (e) $s = 2,\ m = 2$ (f) $s = 2,\ m = 3$

(g) $s = 3,\ m = 1$ (h) $s = 3,\ m = 2$ (i) $s = 3,\ m = 3$

(j) $s = 4,\ m = 1$ (k) $s = 4,\ m = 2$ (l) $s = 4,\ m = 3$

Figure 24 – Number of Qubits vs Depth count for varying $1 \leq s \leq 4$ and $1 \leq s \leq 3$. The curves correspond to isometry decompositions – red represents the Qiskit isometry method and orange represents The Householder Merge Decomposition.

4.3 CHALLENGES IN APPLYING HOUSEHOLDER DECOMPOSITIONS

Despite the advantages discussed in the previous sections, the decomposition method is not without its challenges. The following points highlight limitations and considerations associated with its implementation:

- The Householder decomposition introduces fill-in (additional nonzero values) during the column-by-column decomposition of the isometry, resulting in a denser representation

and increased computational costs. However, for highly sparse and/or small isometries, the amount of fill-in (that is, the numbers that appear in the isometry during the decomposition in places that were originally zero) remains minimal and does not significantly impact the overall decomposition process.

- To better characterize the computational cost of Householder decompositions, a dedicated cost function would need to account for both the sparsity and the number of columns in the isometry. Such an analysis would require additional data to identify consistent patterns and trends. But the execution of the Householder decomposition is computationally intensive due to the substantial number of CNOT gates required., which leads to prolonged execution times, making it impractical for larger datasets without further optimization.

- Since their initial publication and development, the state preparation methods used have undergone optimizations that have reduced their computational costs compared to those presented in the original papers. That said, assuming the existence of a comprehensive cost function that accounts for all relevant variables, it would need to incorporate all three algorithms presented in chapter 3. However, such an analysis is currently constrained by limitations in time and resources.

5 FINAL CONSIDERATIONS

In this work, we performed a comparative analysis of Householder decomposition employing three distinct state preparation methods: Pivot, Merge, and Low Rank. The original work in which this dissertation was inspired (MALVETTI; ITEN; COLBECK, 2021) was extended to evaluate how alternative state preparation strategies—beyond the original Pivot method—affect the efficiency, circuit depth, and resource requirements of sparse isometry decomposition.

For systems with up to six qubits, the state preparation methods exhibit similar behaviors. However, as the isometry becomes denser, Pivot and Merge closely compete to minimize the number of CNOT gates, whereas Low Rank demonstrates instability even for a relatively small number of qubits. In scenarios where the isometry increases in size while preserving sparsity, the experimental data reveal no significant disruptions, and this trend is mirrored in the depth count measurements.

Beyond six qubits, for highly sparse isometries, Merge consistently achieves the lowest CNOT count among the three state preparation methods. Conversely, as the isometry density increases, Low Rank exhibits the steepest growth rate in CNOT usage, followed by Pivot. For larger but sparsity-preserving isometries, the results maintain stability, with minimal variability. Depth count analysis reveals instability in Low Rank for smaller isometries, as indicated by the error bars, while Pivot also demonstrates fluctuations before stabilizing as the system size increases. Merge, in contrast, remains largely unaffected by sparsity variations. When the isometry size increases while maintaining sparsity, Low Rank initially appears unstable but progressively stabilizes as the size grows. Pivot shows a slight peak when the matrix size increases significantly, while Merge remains robust and unaffected.

Based on this analysis, we identify Merge as the most effective state preparation method for the cases examined. Consequently, Merge was selected to compare against Qiskit's isometry function. For smaller systems, up to six qubits, Qiskit's implementation proves superior, requiring fewer CNOT gates and achieving lower depth. However, for larger systems, starting at seven qubits, the Householder Merge method outperforms Qiskit. Our findings indicate that while each method possesses distinct advantages, for highly sparse matrices and matrices with fewer columns, beginning at seven qubits, the techniques for sparse isometries demonstrates superior performance.

As a proposal for future work, a key area of focus should be on reducing both the depth and the number of CNOT gates in the base cases, as these factors directly influence the overall efficiency of the quantum circuit. Several strategies can be pursued to achieve this goal, some of which can be immediately implemented.

One promising path involves leveraging recent advancements in the implementation of multi-controlled gates. Specifically, the work by (NIE; ZI; SUN, 2024) presents a more efficient method for realizing multi-controlled Toffoli and unitary gates by utilizing a combination of a

single clean ancilla and what is referred to as "conditionally clean" ancillas—qubits that are already involved in the circuit. This approach significantly reduces the overhead associated with the ancilla qubits. To incorporate this technique, modifications would need to be made to the existing framework, particularly to the function presented in 1, ensuring that the new gate implementation can be integrated into the existing quantum circuit.

Additionally, optimization techniques from (MALVETTI; ITEN; COLBECK, 2021) could be applied to further improve the circuit's efficiency. This work introduces methods aimed at minimizing the "fill-in" during the decomposition process, which in turn reduces the number of required CNOT gates. By lowering the number of gates, these techniques make the isometry representation of the quantum circuit less dense, thereby improving its overall performance. Implementing these optimizations would not only reduce gate count but also streamline the decomposition process.

Another critical aspect of improving efficiency involves optimizing the underlying code to reduce memory consumption. Given the resource-intensive nature of quantum computations, especially when dealing with large circuits, memory requirements can grow quickly, leading to significant slowdowns. By refining memory management strategies, the algorithm can run more efficiently, with faster execution times and less risk of encountering memory bottlenecks. These optimizations would ensure that the quantum circuit can scale more effectively, allowing for more complex operations to be performed within practical resource limits.

Additionally, further analysis of cost functions is needed to better evaluate trade-offs between gate count, depth, and resource utilization. Investigating hybrid approaches that combine features of different state preparation methods may also yield promising results. Furthermore, extending this work to incorporate noise-aware optimization techniques and fault-tolerant quantum error correction strategies could enhance the practical applicability of the methods in realistic quantum computing scenarios.

In summary, incorporating state-of-the-art techniques for multi-controlled gate implementation, applying optimization strategies to minimize fill-in during decomposition, and improving memory management are key steps that could substantially enhance the efficiency of quantum circuits in future work. These improvements would not only reduce the overall gate count but also accelerate the execution of the algorithm, making it more feasible for real-world quantum computing applications.

REFERENCES

AARONSON, S. Read the fine print. **Nature Physics**, v. 11, n. 4, p. 291–293, 2015.

ALEKSANDROWICZ, G. et al. **Qiskit: An Open-source Framework for Quantum Computing**. 2019.

ARAUJO, I. F. et al. Low-rank quantum state preparation. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, 2023.

ARUTE, F.; AL. et. Quantum supremacy using a programmable superconducting processor. **Nature**, v. 574, n. 7779, p. 505–510, 2019. ISSN 1476-4687. Disponível em: <https://doi.org/10.1038/s41586-019-1666-5>.

BENEDETTI, M. et al. Parameterized quantum circuits as machine learning models. **Quantum Science and Technology**, IOP Publishing, v. 4, n. 4, p. 043001, 2019.

BENIOFF, P. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. **Journal of Statistical Physics**, v. 22, n. 5, p. 563–591, May 1980. ISSN 1572-9613. Disponível em: <https://doi.org/10.1007/BF01011339>.

BIAMONTE, J. et al. Quantum machine learning. **Nature**, Nature Publishing Group, v. 549, n. 7671, p. 195–202, 2017.

CORMEN, T. H. et al. **Introduction to Algorithms**. 3rd. ed. Cambridge, MA: MIT Press, 2009.

CORTESE, J. A.; BRAJE, T. M. **Loading Classical Data into a Quantum Computer**. 2018. Disponível em: <https://arxiv.org/abs/1803.01958>.

COXETER, H. S. M. **Introduction to Geometry**. Second. [S.l.]: Wiley, 1969. ISBN 9780471504580.

DEUTSCH, D. Quantum theory, the church-turing principle and the universal quantum computer. **Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences**, The Royal Society, v. 400, n. 1818, p. 97–117, 1985. ISSN 0080-4630. Disponível em: <http://www.jstor.org/stable/2397601>.

DIVINCENZO, D. P. The physical implementation of quantum computation. **Fortschritte der Physik: Progress of Physics**, Wiley Online Library, v. 48, n. 9-11, p. 771–783, 2000.

EISBERG, R.; RESNICK, R. **Quantum Physics of Atoms, Molecules, Solids, Nuclei, and Particles**. 2nd. ed. New York: Wiley, 1985.

FEYNMAN, R. P. Simulating physics with computers. **International Journal of Theoretical Physics**, v. 21, n. 6, p. 467–488, June 1982. ISSN 1572-9575. Disponível em: <https://doi.org/10.1007/BF02650179>.

GIDNEY, C.; EKERå, M. How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. **Quantum**, Verein zur Forderung des Open Access Publizierens in den Quantenwissenschaften, v. 5, p. 433, abr. 2021. ISSN 2521-327X. Disponível em: <http://dx.doi.org/10.22331/q-2021-04-15-433>.

GLEINIG, N.; HOEFLER, T. An efficient algorithm for sparse quantum state preparation. In: IEEE. **2021 58th ACM/IEEE Design Automation Conference (DAC)**. [S.l.], 2021. p. 433–438.

GROVER, L. K. A fast quantum mechanical algorithm for database search. In: **Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)**. [S.l.: s.n.], 1996. p. 212–219.

GROVER, L. K. Quantum mechanics helps in searching for a needle in a haystack. **Physical review letters**, APS, v. 79, n. 2, p. 325, 1997.

HARROW, A. W.; HASSIDIM, A.; LLOYD, S. Quantum algorithm for linear systems of equations. **Physical review letters**, APS, v. 103, n. 15, p. 150502, 2009.

HOUSEHOLDER, A. S. **Unitary Triangularization of a Nonsymmetric Matrix**. [S.l.]: ACM, 1958. v. 5. 339–342 p.

HU, Z. et al. A general quantum algorithm for open quantum dynamics demonstrated with the Fenna-Matthews-Olson complex. **Quantum**, Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften, v. 6, p. 726, maio 2022. ISSN 2521-327X. Disponível em: <https://doi.org/10.22331/q-2022-05-30-726>.

ITEN, R. et al. Quantum circuits for isometries. **Physical Review A**, American Physical Society (APS), v. 93, n. 3, mar. 2016. ISSN 2469-9934. Disponível em: <http://dx.doi.org/10.1103/PhysRevA.93.032318>.

LEYMANN, F.; BARZEN, J. The bitter truth about gate-based quantum algorithms in the nisq era. **Quantum Science and Technology**, IOP Publishing, v. 5, n. 4, p. 044007, set. 2020. ISSN 2058-9565. Disponível em: <http://dx.doi.org/10.1088/2058-9565/abae7d>.

LLOYD, S.; MOHSENI, M.; REBENTROST, P. **Quantum algorithms for supervised and unsupervised machine learning**. 2013. Disponível em: <https://arxiv.org/abs/1307.0411>.

MACKENZIE, C. E. **Coded Character Sets: History and Development**. [S.l.]: Addison-Wesley, 1980. ISBN 0-201-14460-3.

MALVETTI, E.; ITEN, R.; COLBECK, R. Quantum circuits for sparse isometries. **Quantum**, Verein zur Forderung des Open Access Publizierens in den Quantenwissenschaften, v. 5, p. 412, mar. 2021. ISSN 2521-327X. Disponível em: <http://dx.doi.org/10.22331/q-2021-03-15-412>.

MEZZADRI, F. **How to generate random matrices from the classical compact groups**. 2007. Disponível em: <https://arxiv.org/abs/math-ph/0609050>.

MOTTONEN, M. et al. Transformation of quantum states using uniformly controlled rotations. **arXiv preprint quant-ph/0407010**, 2004.

NEUMANN, N.; PHILLIPSON, F.; VERSLUIS, R. Machine learning in the quantum era. **Digitale Welt**, eMedia Gesellschaft für Elektronische Medien mbH, v. 3, p. 24–29, 2019.

NIE, J.; ZI, W.; SUN, X. **Quantum circuit for multi-qubit Toffoli gate with optimal resource**. 2024. Disponível em: <https://arxiv.org/abs/2402.05053>.

NIELSEN, M. A.; CHUANG, I. L. **Quantum computation and Quantum information**. [S.l.]: Cambridge University Press India, 2000.

OLIPHANT, T. **Guide to NumPy**. [S.l.: s.n.], 2006.

OLIVER, M. **Dream Work**. [S.l.]: Boston: Atlantic Monthly Press, 1986. Poem: *Wild Geese*.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PLESCH, M.; BRUKNER, Č. Quantum-state preparation with universal gate decompositions. **Physical Review A**, APS, v. 83, n. 3, p. 032302, 2011.

PRESKILL, J. Quantum computing in the nisq era and beyond. **Quantum**, Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften, v. 2, p. 79, 2018.

RATH, M.; DATE, H. Quantum data encoding: a comparative analysis of classical-to-quantum mapping techniques and their impact on machine learning accuracy. **EPJ Quantum Technology**, v. 11, n. 1, p. 72, 2024. Accessed April 2025. Disponível em: <https://doi.org/10.1140/epjqt/s40507-024-00285-3>.

REBENTROST, P.; MOHSENI, M.; LLOYD, S. Quantum support vector machine for big data classification. **Physical Review Letters**, American Physical Society (APS), v. 113, n. 13, set. 2014. ISSN 1079-7114. Disponível em: <http://dx.doi.org/10.1103/PhysRevLett.113.130503>.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 21, n. 2, p. 120–126, fev. 1978. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/359340.359342>.

RøNNOW, T. F. et al. Defining and detecting quantum speedup. **Science**, American Association for the Advancement of Science (AAAS), v. 345, n. 6195, p. 420–424, jul. 2014. ISSN 1095-9203. Disponível em: <http://dx.doi.org/10.1126/science.1252319>.

SAKURAI, J. J.; NAPOLITANO, J. **Modern Quantum Mechanics**. 2nd. ed. [S.l.]: Cambridge University Press, 2020.

SCHNEIDER, J.; SMALLEY, I. **What is Quantum Computing?** 2024. Accessed: 2024-11-02. Disponível em: <https://www.ibm.com/topics/quantum-computing>.

SCHULD, M.; PETRUCCIONE, F. **Supervised learning with quantum computers**. [S.l.]: Springer, 2018. v. 17.

SCHULD, M.; SINAYSKIY, I.; PETRUCCIONE, F. An introduction to quantum machine learning. **Contemporary Physics**, Informa UK Limited, v. 56, n. 2, p. 172–185, out. 2014. ISSN 1366-5812. Disponível em: <http://dx.doi.org/10.1080/00107514.2014.964942>.

SCHUMACHER, B. Quantum coding. **Phys. Rev. A**, American Physical Society, v. 51, p. 2738–2747, Apr 1995. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevA.51.2738>.

SHENDE, V. V.; BULLOCK, S. S.; MARKOV, I. L. Synthesis of quantum logic circuits. In: **Proceedings of the 2005 Asia and South Pacific Design Automation Conference**. Shanghai, China: [s.n.], 2005. p. 272–275.

SHOR, P. W. Algorithms for quantum computation: Discrete logarithms and factoring. In: **Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)**. [S.l.: s.n.], 1994. p. 124–134.

SILVA, A. J. da; PARK, D. K. Linear-depth quantum circuits for multiqubit controlled gates. **Physical Review A**, American Physical Society (APS), v. 106, n. 4, out. 2022. ISSN 2469-9934. Disponível em: <http://dx.doi.org/10.1103/PhysRevA.106.042602>.

TANG, E. Quantum principal component analysis only achieves an exponential speedup because of its state preparation assumptions. **Physical Review Letters**, APS, v. 127, n. 6, p. 060503, 2021.

TOFFOLI, T. Reversible computing. In: SPRINGER. **International colloquium on automata, languages, and programming**. [S.l.], 1980. p. 632–644.

VENTURA, D.; MARTINEZ, T. Initializing the amplitude distribution of a quantum state. **Foundations of Physics Letters**, Springer, v. 12, n. 6, p. 547–559, 1999.

VERAS, T. M. de; SILVA, L. D. da; SILVA, A. J. da. Double sparse quantum state preparation. **Quantum Information Processing**, Springer, v. 21, n. 6, p. 1–13, 2022.

VERAS, T. M. L. de et al. Circuit-based quantum random access memory for classical data with continuous amplitudes. **IEEE Transactions on Computers**, v. 70, n. 12, p. 2125–2135, 2021.

WEIGOLD, M. et al. Encoding patterns for quantum algorithms. **IET Quantum Communication**, Wiley Online Library, v. 2, n. 4, p. 141–152, 2021.

WEISSTEIN, E. W. **Isometry**. n.d. <https://mathworld.wolfram.com/Isometry.html>. Accessed: 2024-10-23.

WHURLEY; SMITH, F. E. **Quantum Computing For Dummies**. [S.l.]: For Dummies, 2023. ISBN 978-1-119-93390-8.

YANOFSKY, N. S.; MANNUCCI, M. A.; MANNUCCI, M. A. **Quantum computing for computer scientists**. [S.l.]: Cambridge University Press Cambridge, 2008. v. 20.

ZHANG, X.-M.; YUNG, M.-H.; YUAN, X. **Low-depth Quantum State Preparation**. 2021. Disponível em: <https://arxiv.org/abs/2102.07533>.

ZHANG, Y. et al. Quantum simulation of the radical pair dynamics of the avian compass. **The Journal of Physical Chemistry Letters**, American Chemical Society (ACS), v. 14, n. 3, p. 832–837, jan. 2023. ISSN 1948-7185. Disponível em: <http://dx.doi.org/10.1021/acs.jpclett.2c03617>.

APPENDIX A

MERGE STATE PREPARATION

Let us illustrate the operation of the Merge State Preparation on a sparse quantum state through a detailed example.

Example 1 *Given an initial state $|v\rangle$, we want to reach a final state $|u\rangle$:*

$$|v\rangle = \frac{1}{\sqrt{14}}(|001\rangle + 2\,|100\rangle + 3\,|111\rangle) \quad \rightarrow \quad |u\rangle = |000\rangle$$

Algorithm 1

Step 1. *Since the state $|v\rangle$ has $n = 3$ qubits, we initialize an empty circuit C:*

$$C = |0\rangle^{\otimes n} = |000\rangle$$

We define the set S that contains the indices of the non-zero amplitudes in the target state:

$$S = \{001, 100, 111\} \quad \rightarrow \quad |\,S\,| = 3$$

Step 2. *We select a qubit $b \in \{1, 2, 3\}$ in a way that satisfies:*

$$S_0 = \{x \in S \mid x[b] = 0\}; \quad S_1 = \{x \in S \mid x[b] = 1\}$$

Let's analyze the cases individually.

- $b = 1$ *(qubit in the first position)*

$$S_0 = \{001\}$$

$$S_1 = \{100, 111\}$$

- $b = 2$ *(qubit in the second position)*

$$S_0 = \{001, 100\}$$

$$S_1 = \{111\}$$

- $b = 3$ *(qubit in the third position)*

$$S_0 = \{100\}$$

$$S_1 = \{001, 111\}$$

Since we want to merge the states of S_1 into S_0, we choose b in a way that minimizes the amount of states in S_1. In this case, we choose $b = 2$.

Step 3. *To select the basis states for merging, we choose the ones that are most similar, which means those with the fewest differing qubits. For example, the states $|001\rangle$ and $|111\rangle$ differ in two qubits (qubits 1 and 2), while $|100\rangle$ and $|111\rangle$ also differ in two qubits (qubits 2 and 3). Since both pairs differ by the same number of qubits, we choose the first pair in increasing numerical order.*

Then, the basis states $|\phi\rangle$ we want to merge are:

$$|\phi\rangle = \frac{1}{\sqrt{14}}(|001\rangle + 3\,|111\rangle)$$

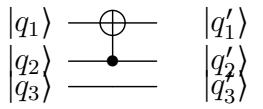
Calculating the angle ω of the merging states:

$$\tan(\omega) = \frac{|1/\sqrt{14}|}{|3/\sqrt{14}|} = \frac{1}{3} \rightarrow \omega = \arctan\left(\frac{1}{3}\right)$$

The relative phase between these states is:

$$\alpha = Arg(3) - Arg(1) = 0$$

Step 4. *We want to merge the states $|001\rangle$ and $|111\rangle$ by the application of CNOT gates in such way that the control qubit is the qubit $b = 2$.*



The state becomes:

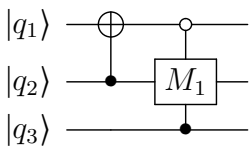
$$|v'\rangle = \frac{1}{\sqrt{14}}(|001\rangle + 2\,|100\rangle + 3\,|011\rangle)$$

Step 5. *Now we use M gate to "merge" the amplitudes of these states. Remember, M is defined as:*

$$M = \begin{bmatrix} sin(\omega) & e^{i\alpha}cos(\omega) \\ e^{-i\alpha}cos(\omega) & -sin(\omega) \end{bmatrix} \rightarrow M_1 = \begin{bmatrix} \frac{1}{\sqrt{10}} & \frac{3}{\sqrt{10}} \\ \frac{3}{\sqrt{10}} & -\frac{1}{\sqrt{10}} \end{bmatrix}$$

We apply the matrix M_1 to the differing qubit $b = 2$. The matrix M_1 is controlled by the other qubits, which should all have the same value at this stage, determining if the control is open or closed.

In our example, qubits 1 and 3 in the merging states have values $|0\rangle$ and $|1\rangle$, respectively. Since we only want to merge these basis states, we set the control for qubit 1 to open and the control for qubit 3 to closed. The circuit becomes:



We can verify that the final state is:

$$\left|\tilde{v'}\right\rangle = \frac{1}{\sqrt{14}}(\sqrt{10}\,|001\rangle + 2\,|100\rangle)$$

Step 6. *Now that the first merge is complete, we replace the set S with our resulting set after the merge:*

$$S = \{001, 100\}$$

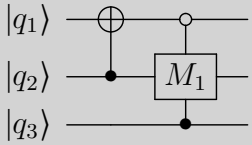
Step 7. *Return the circuit C.*

With the first merge completed, the Merge State Preparation proceeds by applying Algorithm 2 to begin constructing the desired circuit. The procedure is as follows:

Algorithm 2

Step 1. *Start with an initial empty quantum circuit C and define a set containing the basis states $S = \{001, 100, 111\}$ of the target state $|v\rangle$.*

Step 2. *Since $|\,S\,| > 1$, we use algorithm 1, we find the circuit C':*



Step 3. *We update the state as: $|v\rangle = C'\,|v\rangle = \frac{1}{\sqrt{14}}(\sqrt{10}\,|001\rangle + 2\,|100\rangle)$*

Step 4. *We update the quantum circuit as: $C = C' \cdot C$*

Since the set S has $|\,S\,| = 2$ elements, so we use Algorithm 1 again.

Algorithm 1

Step 1. *Since the state $|v\rangle$ has $n = 3$ qubits, we initialize an empty circuit $|\psi\rangle$:*

$$|\psi\rangle = |0\rangle^{\otimes n} = |000\rangle$$

We define the set S that contains the indices of the non-zero amplitudes in the target size:

$$S = \{001, 100\} \quad \rightarrow \quad |\,S\,| = 2$$

Step 2. *We select a qubit $b \in \{1, 2\}$ in a way that satisfies:*

$$S_0 = \{x \in S \mid x[b] = 0\}; \quad S_1 = \{x \in S \mid x[b] = 1\}$$

Let's analyze the cases individually.

- $b = 1$ *(qubit in the first position)*

$$S_0 = \{001\}$$

$$S_1 = \{100\}$$

- $b = 2$ *(qubit in the second position)*

$$S_0 = \{001, 100\}$$

$$S_1 = \varnothing$$

- $b = 3$ *(qubit in the third position)*

$$S_0 = \{100\}$$

$$S_1 = \{001\}$$

We choose $b = 1$.

Step 3. *The basis states* $|\phi\rangle$ *we want to merge are:*

$$|\phi\rangle = \frac{1}{\sqrt{14}}(\sqrt{10}\,|001\rangle + 2\,|100\rangle)$$

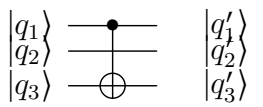
Calculating the angle ω *of the chosen states:*

$$\tan(\omega) = \frac{|\sqrt{10}/\sqrt{14}|}{|2/\sqrt{14}|} = \frac{\sqrt{10}}{2} \rightarrow \omega = \arctan\left(\frac{\sqrt{10}}{2}\right) \qquad \text{(APPENDIX A.1)}$$

The relative phase between these states is:

$$\alpha = Arg(2) - Arg(\sqrt{10}) = 0$$

Step 4. *We want to merge the states* $|001\rangle$ *and* $|100\rangle$ *by the application of CNOT gates in such way that the control qubit is the qubit* $b = 1$.



The state becomes:

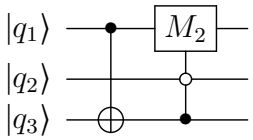
$$|v'\rangle = \frac{1}{\sqrt{14}}(\sqrt{10}\,|001\rangle + 2\,|101\rangle)$$

Step 5. *Now we use M gate to "merge" the amplitudes of these states. Remember, M is defined as:*

$$M = \begin{bmatrix} sin(\omega) & e^{i\alpha}cos(\omega) \\ e^{-i\alpha}cos(\omega) & -sin(\omega) \end{bmatrix} = \begin{bmatrix} \sqrt{\frac{5}{7}} & \sqrt{\frac{2}{7}} \\ \sqrt{\frac{2}{7}} & -\sqrt{\frac{5}{7}} \end{bmatrix}$$

We apply the matrix M to the differing qubit $b = 1$. The matrix M is controlled at the other qubits, which should be at the same value at this point, determining if the control is open or closed.

Now, qubit 2 is open and qubit 3 is closed. The circuit becomes:



We can verify that the final state is:

$$\left|\tilde{v'}\right\rangle = |001\rangle$$

Step 6. *Now that the second merge is complete, we replace the set S with our resulting set after the merge:*

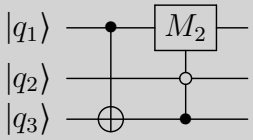
$$S = \{001\}$$

Step 7. *Return the circuit C.*

Algorithm 2

Step 1. *Start with an initial empty quantum circuit C.*

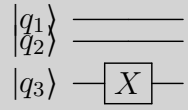
Step 2. *After Algorithm 1, we find the circuit C':*



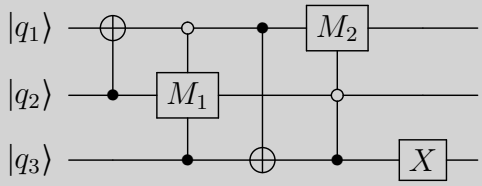
Step 3. *We update the state as: $|v\rangle = C'\,|v\rangle = |001\rangle$*

Step 4. *We update the quantum circuit as: $C = C' \cdot C$*

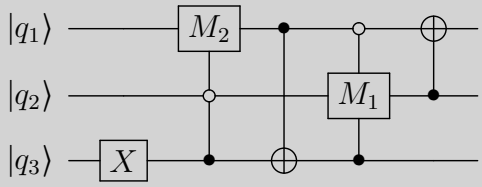
Step 5. *Now, the set S has $|\,S\,| = 1$ element. So, we apply NOT gates to transform the state to a zero state:*



The circuit C becomes:



Step 6. *Invert the gates from C and reverse their order, so C becomes:*



Step 7. *Return the final circuit C.*

APPENDIX B

CVO-QRAM STATE PREPARATION

Let us illustrate the operation of the CVO-QRAM State Preparation on a sparse quantum state through a detailed example.

Example 2 *Given an initial state* $|v\rangle$*:*

$$|v\rangle = \frac{1}{\sqrt{2}}(|001\rangle + |101\rangle)$$

Step 1. *Start with an initial state* $|\psi\rangle = |u\rangle\,|m\rangle$*, where* $|u\rangle$ *is an ancilla initialized in the state* $|1\rangle$ *and* $|m\rangle$ *is a memory register initialized in the state* $|0\rangle^{\otimes n}$*, where* n *is the number of qubits of the system. That gives us:*

$$|u\rangle = |1\rangle\,; \quad |m\rangle = |0\rangle^{\otimes 3} = |000\rangle$$

So, we get the initial state $|\psi_0\rangle$*:*

$$|\psi_0\rangle = |u\rangle\,|m\rangle = |1\rangle\,|000\rangle$$

Step 2. *We sort the input patterns in ascending order of the number of bits with value 1.*

Here:

- $x_0 = 001$
- $x_1 = 101$

Thus, the sorted order is:

$$\textit{data} = \{(001, \frac{1}{\sqrt{2}}), (101, \frac{1}{\sqrt{2}})\}$$

Now we enter a loop that computes each pair (x, c_x)*, following the algorithm.*

Computing the first pair $(001, \frac{1}{\sqrt{2}})$

Step 3. *For each set* (x, c_x)*:*

a) Compute the number of bits with value 1 *in a variable* t *and identify and compute a list containing the positions of the input patterns* x *where the processing term is equal to* 1*.*

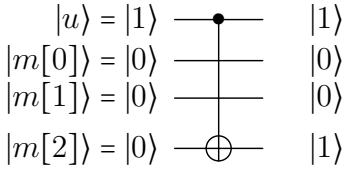
- *Number of bits with value* 1*:* $t = 1$

- *Position of the bits with value* 1*:* $l = [2]$

b) *To compute the desired basis state, apply CNOT operations controlled at the ancilla and targeting the memory register* $|m\rangle$ *where it has value* 1*, that is:*

$$\begin{aligned} |\psi_1\rangle &= \Pi_{l_i \epsilon l} CNOT_{(u,m[l_i])} |\psi_0\rangle \\ &= CNOT_{(1,m[2])} |\psi_0\rangle \end{aligned}$$

In other words, we need CNOT operations controlled in the qubit $|u\rangle$ *and target at* $|m[2]\rangle$*:*



The state becomes:

$$|\psi_1\rangle = |1\rangle |001\rangle$$

c) *Initialize the amplitudes by applying the rotation operator* $U(x_0, \gamma_0)$ *targeting* $|u\rangle$*, controlled by the qubits where* $x[l_i] = 1$*, with* $0 \leq i \leq t-1$*. We apply:*

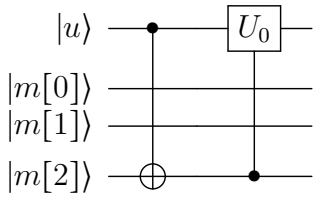
$$\begin{aligned} |\psi_2\rangle &= C^t U_{(m[l_0,l_1,\cdots,l_{t-1}],u)} |\psi_1\rangle \\ &= C^1 U_{(m[2],u)} |\psi_1\rangle \end{aligned}$$

By definition, $\gamma_0 = 1$*, and based on our data,* $x_0 = \frac{1}{\sqrt{2}}$*.*

Then:

$$U_0 = \begin{bmatrix} \sqrt{\frac{\gamma_0 - |x_0|^2}{\gamma_0}} & \frac{x_0}{\sqrt{\gamma_0}} \\ \frac{-x_0 *}{\sqrt{\gamma_0}} & \sqrt{\frac{\gamma_0 - |x_0|^2}{\gamma_0}} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

Applying this rotation to the ancillary qubit $|u\rangle$ *controlled by* $|m[2]\rangle$*, the circuit becomes:*



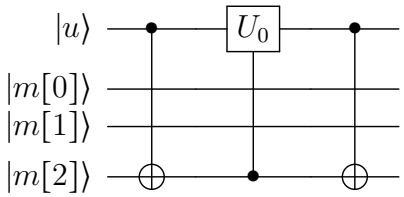
The state becomes:

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,|001\rangle$$

d) If not in the final iteration, apply $CNOT$ gates to restore the $|m\rangle$ register. Like we did in step 4:

$$\begin{aligned}|\psi_3\rangle &= \Pi_{l_i \epsilon l} CNOT_{(u,m[l_i])}\,|\psi_2\rangle \\ &= CNOT_{(1,m[2])}\,|\psi_2\rangle\end{aligned}$$

The circuit becomes:



The state becomes:

$$|\psi_3\rangle = \frac{1}{\sqrt{2}}\,|0\rangle\,|001\rangle + \frac{1}{\sqrt{2}}\,|1\rangle\,|000\rangle$$

Now that the first basis state is complete, let's compute the second.

Computing the second pair $(101, \frac{1}{\sqrt{2}})$

Step 3. *For each set (x, c_x):*

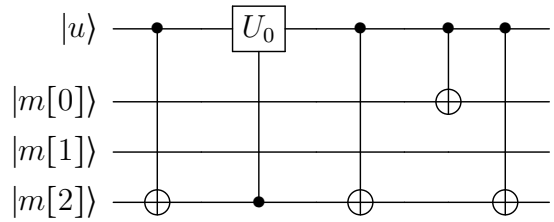
a) We compute the number of bits with value 1 in a variable t and identify and compute a list containing the positions of the input patterns x where the processing term is equal to 1.

- *Number of bits with value 1: $t = 2$*
- *Position of the bits with value 1: $l = [0, 2]$*

b) To compute the desired basis state, apply CNOT operations controlled at the ancilla $|u\rangle$ and targeting the memory register $|m\rangle$ where it has value 1, that is:

$$\begin{aligned}|\psi_4\rangle &= \Pi_{l_i \in l} CNOT_{(u,m[l_i])} |\psi_3\rangle \\ &= CNOT_{(u,m[0])} CNOT_{(u,m[2])} |\psi_3\rangle\end{aligned}$$

Giving us the circuit:



The state becomes:

$$|\psi_4\rangle = \frac{1}{\sqrt{2}} |0\rangle |001\rangle + \frac{1}{\sqrt{2}} |1\rangle |101\rangle$$

c) Applying the rotation operator $U(x_1, \gamma_1)$ targeting $|u\rangle$, controlled by the qubits where $x[l_i] = 1$, with $0 \leq i \leq t-1$. We apply:

$$\begin{aligned}|\psi_5\rangle &= C^t U_{(m[l_0,l_1,\cdots,l_{t-1}],u)} |\psi_4\rangle \\ &= C^1 U_{(m[0,2],u)} |\psi_4\rangle\end{aligned}$$

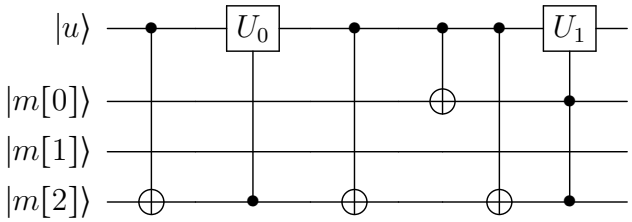
Calculating γ_1:

$$\gamma_1 = \gamma_0 - |\, x_0 \,|^2 = 1 - |\, \frac{1}{\sqrt{2}} \,|^2 = \frac{1}{2}$$

Then:

$$U_1 = \begin{bmatrix} \sqrt{\frac{\gamma_1 - |x_1|^2}{\gamma_1}} & \frac{x_1}{\sqrt{\gamma_1}} \\ \frac{-x_1*}{\sqrt{\gamma_1}} & \sqrt{\frac{\gamma_1 - |x_1|^2}{\gamma_1}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

The circuit becomes:



The state becomes:

$$|\psi_5\rangle = \frac{1}{\sqrt{2}}|0\rangle|001\rangle + \frac{1}{\sqrt{2}}|0\rangle|101\rangle$$

d) Since we are in the final step, there's no need to reverse any $CNOT$ gates applied.

Step 4. *After loading both patterns, the state is:*

$$|\psi_5\rangle = \frac{1}{\sqrt{2}}|0\rangle|001\rangle + \frac{1}{\sqrt{2}}|0\rangle|101\rangle$$

Since the ancilla qubit $|u\rangle$ is in state $|0\rangle$, we can ignore it, and the final state is:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|001\rangle + \frac{1}{\sqrt{2}}|101\rangle \square$$

With that, $|\psi\rangle = |v\rangle$ is the state we wanted to prepare.

APPENDIX C

PIVOT STATE PREPARATION

Let us illustrate the operation of the Pivot State Preparation on a sparse quantum state through a detailed example.

Example 3 *Given an initial state $|v\rangle$, we want to reach a final state $|u\rangle$:*

$$|v\rangle = \frac{1}{\sqrt{2}}|011\rangle + \frac{1}{\sqrt{2}}|101\rangle \quad \rightarrow \quad |u\rangle = \frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|001\rangle$$

Note that this state has 3 qubits, which gives us $2^3 = 8$ possible basis states. Notice that out of these 8, only 2 of them have nonzero amplitude. In other words, this state is sparse and $nnz(v) = 2$. From equation 2.20:

$$s = log_2 2 \quad \rightarrow \quad s = 1$$

Block T is the block where all nonzero entries are grouped. It is determined by the values of the first $n - s$ qubits (for example, $|0\rangle^{\otimes(n-s)}$ by default, but this is flexible).

Step 1.

Step 2. *We select a component outside the block T whose amplitude is different from zero and rewrite it as:*

- *Outside the block T:*

$$|011\rangle \quad \rightarrow \quad |01\rangle|1\rangle = |t'\rangle^{\otimes n-s}|r'\rangle^{\otimes s}$$

Now we select a component inside block T whose amplitude has a zero value and rewrite it as):

- *Inside block T:*

$$|000\rangle \quad \rightarrow \quad |00\rangle|0\rangle = |t\rangle^{\otimes n-s}|r\rangle^{\otimes s}$$

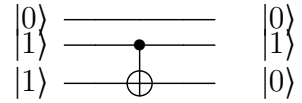
Step 3. *Choose a qubit where $|t'\rangle^{\otimes n-s}$ and $|t\rangle^{\otimes n-s}$ differ. In our case, we select the second qubit (from left to right) as shown in the equation below.*

$$|t'\rangle^{\otimes n-s} = \left|0\,\boxed{1}\right\rangle \qquad \text{and} \qquad \left|0\,\boxed{0}\right\rangle$$

Step 4. *With $|t'\rangle^{\otimes n-s}$ as the control qubit, we will use at most $n-1 = 3-1 = 2$ CNOT gates to modify $|t'\rangle^{\otimes n-s}|r'\rangle^{\otimes s}$ such that $|r'\rangle^{\otimes s} = |r\rangle^{\otimes s}$. That is,*

$$|t'\rangle^{\otimes n-s}|r'\rangle^{\otimes s} = |01\rangle|1\rangle \quad \rightarrow \quad |01\rangle|0\rangle = |t''\rangle^{\otimes n-s}|r\rangle^{\otimes s}$$

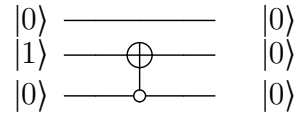
The circuit becomes:



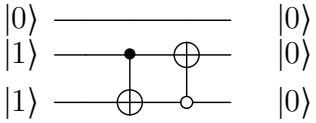
Step 5. *Following a similar approach to the previous step, apply an s-controlled NOT gate to modify the circuit such that*

$$|t''\rangle^{\otimes n-s}|r\rangle^{\otimes s} = |01\rangle|0\rangle \quad \rightarrow \quad |00\rangle|0\rangle = |t\rangle^{\otimes n-s}|r\rangle^{\otimes s}$$

The circuit becomes:



By combining the previous circuits, we obtain circuit C_1:



Note that none of the other components of the initial state should be affected in this process. Applying circuit C_1 to state $|v\rangle$, we obtain a new state:

$$|v_1\rangle = C_1|v\rangle = \frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|101\rangle$$

Step 6. *Since not all nonzero terms are in block T, return to step 1.*

Now we repeat the process of the algorithm to the state $|v_1\rangle$.

Step 1.

Step 2. *We select a component outside block T with nonzero amplitude and rewrite it as:*

- *Outside block T:*

$$|101\rangle \quad \rightarrow \quad |10\rangle|1\rangle = |t'\rangle^{\otimes n-s}|r'\rangle^{\otimes s}$$

Now we select a component inside block T with zero amplitude and rewrite it as:

- *Inside block T:*

$$|001\rangle \quad \rightarrow \quad |00\rangle|1\rangle = |t\rangle^{\otimes n-s}|r\rangle^{\otimes s}$$

Step 3. *Choose a qubit where $|t'\rangle^{\otimes n-s}$ and $|t\rangle^{\otimes n-s}$ differ. In our case, we select the first qubit as shown in the equation below.*

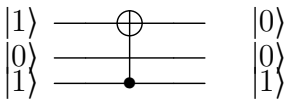
$$|t'\rangle^{\otimes n-s} = \big|\,1\,0\big\rangle \qquad \text{and} \qquad \big|\,0\,0\big\rangle$$

Step 4. *Since $|r'\rangle^{\otimes s}$ and $|r\rangle^{\otimes s}$ are already equal, no action is required.*

Step 5. *Apply an s-controlled NOT gate to adjust the circuit so that*

$$|t'\rangle^{\otimes n-s}\,|r\rangle^{\otimes s} = |10\rangle\,|1\rangle \quad \rightarrow \quad |00\rangle\,|1\rangle = |t\rangle^{\otimes n-s}\,|r\rangle^{\otimes s}$$

This can be achieved through the application of circuit C_2:



Note that none of the other components of the initial state should be affected in this process. Applying circuit C_2 to the state $|v_1\rangle$ yields a new state:

$$|u\rangle = C_2\,|v_1\rangle = C_2(C_1\,|v\rangle) = \frac{1}{\sqrt{2}}\,|000\rangle + \frac{1}{\sqrt{2}}\,|001\rangle$$

Step 6. *Since all nonzero terms are now in block T, we finish the algorithm.*
We can also express state $|u\rangle$ as:

$$|u\rangle = |00\rangle \otimes (\frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle)$$

The circuit $C_2C_1 = Piv_v$ is exactly what we sought, as seen in Equation 2.20.

APPENDIX D

LOW-RANK STATE PREPARATION

Let us illustrate the operation of the Low-Rank State Preparation on a sparse quantum state through a detailed example.

Example 4 *Given an initial state* $|v\rangle$*:*

$$|v\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Step 1. *The Schmidt Decomposition is done in a classical regime. For the state* $|\psi\rangle$*, this results in:*

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle_A|0\rangle_B + \frac{1}{\sqrt{2}}|1\rangle_A|1\rangle_B$$

The Schmidt values associated are:

- *Schmidt rank* $k = 2 \quad \rightarrow \quad m = \lceil log_2(2) \rceil = 1$
- *Schmidt coefficients* $\sigma_1 = \sigma_2 = \frac{1}{\sqrt{2}}$
- *Schmidt basis:* $\{|0\rangle_A, |1\rangle_A\}$ *and* $\{|0\rangle_B, |1\rangle_B\}$

The demonstration of the calculations is left to the reader.

Step 2. *The system is initialized in the zero state with two qubits, that is,* $|0\rangle|0\rangle$*. Then, we initialize a quantum state in the first register encoding Schmidt coefficients as:*

$$\sum_i \sigma_i |i\rangle|0\rangle = \frac{1}{\sqrt{2}}|0\rangle|0\rangle + \frac{1}{\sqrt{2}}|1\rangle|0\rangle$$

In the circuit, this can be done by applying a Hadamard gate to the first qubit:

$$\begin{array}{lll} q_A = |0\rangle & -\boxed{H}- & \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \\ q_B = |0\rangle & ------ & |0\rangle \end{array}$$

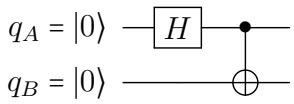
Now the state is:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A + |1\rangle_A) \otimes |0\rangle_B$$

Step 3. *Apply, at max,* $\lfloor n/2 \rfloor = \lfloor 2/2 \rfloor = 1$ *CNOT operations, controlled qubit* A *and targeting qubit* B*, to generate entanglement. In other words, we want to generate:*

$$\sum_i \sigma_i |i\rangle|i\rangle$$

The circuit becomes:



The state becomes:

$$|\psi\rangle = \frac{1}{\sqrt{2}} |0\rangle_A |0\rangle_B + \frac{1}{\sqrt{2}} |1\rangle_A |1\rangle_B$$

Step 4. *Now, we need to apply unitary operations U to the first register and V^T to the second register to map computational basis to the Schmidt basis:*

$$U|i\rangle = |i_A\rangle, \qquad V^T|i\rangle = |i_B\rangle$$

But in our case, the Schmidt basis are already $|0\rangle$ and $|1\rangle$, which are computational basis states. Then, $U = V^T = I$. So, there's no operation needed.